



UNIVERSITY OF  
CAMBRIDGE

# Retrieval of research-level mathematics via joint modelling of text and types

Yiannos Stathopoulos

March, 2022



St. Edmund's College, Cambridge

This dissertation is submitted for the degree of Doctor of Philosophy



# Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. It is not substantially the same as any that I have submitted, or am concurrently submitting, for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my dissertation has already been submitted, or is being concurrently submitted, for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. This dissertation does not exceed the prescribed limit of 60 000 words.

Yiannos Stathopoulos

March, 2022



# Abstract

## Retrieval of research-level mathematics via joint modelling of text and types

*Yiannos Stathopoulos*

Recent work in Mathematical Information Retrieval (MIR) emphasises the retrieval of symbolic formulae but ignores their interactions with the text. However, mathematicians think and communicate ideas that take the form of mathematical objects and structures using both modalities.

An important component of the interaction between the modalities is the mathematical type. Types are technical terms that occur in the textual modality and are used to refer to mathematical ideas.

When linked to the symbolic modality, types assume the role of denotations to mathematical expressions. Existing MIR retrieval models ignore this connection and treat types in the same way as text-based IR systems would: as a bag of independent words, rather than potential multi-word units.

In this thesis I ask two questions: can MIR of research mathematics benefit from the recognition of types in the textual modality, and can it benefit from linking the two modalities by assigning denotations from the textual to the symbolic modality?

To investigate these questions I develop a method for constructing a test collection for research-level mathematics, which relies on observing the MathOverflow online community (this aspect of my work has resulted in the CUMTC collection with 160 queries) and use machine learning to link variables in context to their type. I then develop models that jointly model the modalities by combining type-based textual retrieval with typed formula retrieval. A key idea is typed unification – matching formulae by their structure and by the types of their constituents.

My experiments show that textual types in queries improve textual retrieval significantly over traditional term-based IR methods when queries are expanded with similar types from an embedding space. My best model of this class beats the best commonly available MIR model by a margin of 144% on my test collection (MAP=.173 vs. MAP=.071). However, I was unable to prove the additional benefit of my joint typed models, although I was able to empirically describe some retrieval situations where retrieval benefits from these more complex models.



# Acknowledgements

I acknowledge work done in collaboration with Simon Baker and Marek Rei on my variable typing machine learning task for Chapter 5. Simon Baker helped me develop the natural language annotation scheme for the task, annotated part of the data and developed the convolutional neural network evaluated in Chapter 5. Marek Rei trained his own bidirectional LSTM model with data for the variable typing task, helped with evaluating his model for the task (Chapter 5) and provided predictions for all documents and queries in my test collection. I used Marek's predictions to build the typed retrieval models I evaluate in Chapter 7. I state to the best of my ability what work has been done as part of this collaboration in the main text of my thesis.

I would like to express my enormous gratitude to my supervisor, Professor Simone Teufel. Her patience, dedication and invaluable, detailed feedback in every step of the way have been instrumental to the completion of this PhD thesis.

I am very grateful to my college and department for their support and for making my experience the best I could have ever hoped for. Special thanks to Lise Gough at the Faculty of Computer Science and Technology for her immense patience, support and for always being available to answer my questions.

I am grateful for the help I have received from my friends and collaborators, Marek Rei and Simon Baker. A big thank you to Marek for presenting our 2018 paper in New Orleans. I would also like to thank my office mates, Wenduan Xu and Sandro Bauer, for the many stimulating conversations we have shared and for their friendship throughout the duration of this PhD and beyond. Many thanks to Kris Cao for answering all my mathematical questions and for contributing to my Chapter 4 inter-annotator agreement experiments. Everyone in my college who also participated in the aforementioned experiments – thank you.

I dedicate this thesis to my family: my father, Andreas, mother, Avgi and brother Yiorgos, whose boundless patience, love, support and encouragement carried me to the very end. Thank you.



# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>   | <b>15</b> |
| <b>2</b> | <b>Background and Related Work</b>  | <b>17</b> |
| 2.1      | Overview of Information Retrieval . . . . .                               | 17        |
| 2.2      | Mathematical Information Retrieval . . . . .                              | 22        |
| 2.2.1    | Representation of Mathematical Expressions . . . . .                      | 22        |
| 2.2.2    | Non-structural MIR models . . . . .                                       | 25        |
| 2.2.3    | Tree-based MIR models . . . . .   | 26        |
| 2.2.4    | The MCAT model . . . . .  | 30        |
| 2.2.5    | Tangent . . . . .   | 31        |
| 2.3      | Information Retrieval Evaluation . . . . .                                | 33        |
| 2.3.1    | The Test Collection Paradigm . . . . .                                    | 34        |
| 2.3.2    | The Probability Ranking Principle . . . . .                               | 36        |
| 2.3.3    | Evaluation Measures . . . . .   | 36        |
| 2.4      | Methodologies for Building Test Collections . . . . .                     | 40        |
| 2.4.1    | The Cranfield Methodology . . . . .                                       | 43        |
| 2.4.2    | TREC . . . . .  | 46        |
| 2.4.3    | Yahoo! Chiebukuro . . . . .   | 48        |
| 2.5      | The Language and Discourse of Mathematical Content . . . . .              | 51        |
| 2.6      | Summary and Approach . . . . .  | 54        |
| <b>3</b> | <b>A Test Collection of Research-Level Mathematical Information Needs</b> | <b>57</b> |
| 3.1      | Motivation for a New MathIR Test Collection . . . . .                     | 57        |
| 3.2      | Cambridge University Mathematics Test Collection (CUMTC) . . . . .        | 61        |
| 3.2.1    | Document Collection . . . . .   | 62        |
| 3.2.2    | MathOverflow . . . . .  | 63        |
| 3.2.3    | Queries and Relevance Judgements in an MO context . . . . .               | 66        |
| 3.2.4    | The Construction Process . . . . .  | 72        |
| 3.2.5    | Test Collection Overview and Statistics . . . . .                         | 75        |
| 3.3      | Chapter Summary . . . . .   | 77        |

|          |   |            |
|----------|---|------------|
| <b>4</b> | <b>Mathematical Types for Retrieval</b>                   | <b>79</b>  |
| 4.1      | Definition of Mathematical Types . . . . .                | 82         |
| 4.2      | Automatic Type Detection . . . . .                        | 85         |
| 4.2.1    | Creation of Seed Dictionary . . . . .                     | 85         |
| 4.2.2    | Gold Standard Evaluation of the Seed Dictionary . . . . . | 86         |
| 4.2.3    | The Double Suffix-Trie Algorithm . . . . .                | 88         |
| 4.3      | Type Embeddings . . . . .                                 | 93         |
| 4.4      | Retrieval based on Textual Types . . . . .                | 95         |
| 4.5      | Comparison with Related Work . . . . .                    | 100        |
| 4.6      | Chapter Summary . . . . .                                 | 102        |
| <b>5</b> | <b>Variable Typing</b>                                    | <b>105</b> |
| 5.1      | A Variable Typing Data Set . . . . .                      | 108        |
| 5.1.1    | Sentence Sampling . . . . .                               | 108        |
| 5.1.2    | Annotation Scheme . . . . .                               | 110        |
| 5.1.3    | Human Agreement Experiment . . . . .                      | 111        |
| 5.1.4    | Bulk Corpus Annotation . . . . .                          | 112        |
| 5.2      | Comparison with Related Work . . . . .                    | 112        |
| 5.3      | Experiments . . . . .                                     | 115        |
| 5.3.1    | Models for Variable Typing . . . . .                      | 115        |
| 5.3.2    | Results . . . . .   | 119        |
| 5.4      | Variable Typing for Typed Retrieval . . . . .             | 121        |
| 5.5      | Limitations of Variable Typing . . . . .                  | 123        |
| 5.6      | Chapter Summary . . . . .                                 | 124        |
| <b>6</b> | <b>Typed Joint Retrieval</b>                              | <b>127</b> |
| 6.1      | Matching Operations on SLTs . . . . .                     | 130        |
| 6.1.1    | Extensions to SLTs . . . . .                              | 130        |
| 6.1.2    | Exact SLT Matching . . . . .                              | 134        |
| 6.1.3    | SLT Matching with $\alpha$ -equivalence . . . . .         | 137        |
| 6.1.4    | Recursive SLT Matching . . . . .                          | 144        |
| 6.2      | Unification over SLTs . . . . .                           | 144        |
| 6.3      | Practical Typed MIR . . . . .                             | 147        |
| 6.3.1    | Type Disambiguation . . . . .                             | 148        |
| 6.3.2    | Typing of non-variable SLTs . . . . .                     | 151        |
| 6.3.3    | Scoring Tree Similarity . . . . .                         | 152        |
| 6.4      | Retrieval Models . . . . .                                | 156        |
| 6.4.1    | Tree Matching with Type Unification . . . . .             | 156        |
| 6.4.2    | Tree Matching without Type Unification . . . . .          | 159        |

|          |   |            |
|----------|---|------------|
| 6.4.3    | Typed Tangent Retrieval Models . . . . .                  | 160        |
| 6.5      | Chapter Summary . . . . .                                 | 161        |
| <b>7</b> | <b>Retrieval Experiments</b>                              | <b>163</b> |
| 7.1      | Type-based Textual Retrieval . . . . .                    | 164        |
| 7.1.1    | Experimental Design . . . . .                             | 164        |
| 7.1.2    | Results and Discussion . . . . .                          | 168        |
| 7.2      | Untyped Formula-Based Retrieval . . . . .                 | 171        |
| 7.2.1    | Experimental Design . . . . .                             | 171        |
| 7.2.2    | Results and Post-hoc analyses . . . . .                   | 173        |
| 7.2.2.1  | Performance of UT . . . . .                               | 174        |
| 7.2.2.2  | TM1– vs. TM2– . . . . .                                   | 180        |
| 7.2.3    | Comparison to Textual Models . . . . .                    | 183        |
| 7.3      | Joint Retrieval: Typed Text and Typed Formulae . . . . .  | 184        |
| 7.3.1    | Experimental Design . . . . .                             | 184        |
| 7.3.2    | Results and post-hoc analyses . . . . .                   | 187        |
| 7.3.2.1  | Performance of type unification . . . . .                 | 189        |
| 7.3.2.2  | Performance of semantic component . . . . .               | 194        |
| 7.3.2.3  | Type-aware vs Established MIR System . . . . .            | 198        |
| 7.3.3    | Comparison to Textual Models . . . . .                    | 201        |
| 7.4      | Chapter Summary . . . . .                                 | 202        |
| <b>8</b> | <b>Conclusions</b>  | <b>205</b> |
| 8.1      | Contributions . . . . .                                   | 205        |
| 8.2      | Recent Developments in MIR Test Collections . . . . .     | 207        |
| 8.3      | Limitations and future work . . . . .                     | 208        |
| 8.3.1    | Graph Neural Network Representations . . . . .            | 209        |
| 8.3.2    | Typed Tangent with Unification . . . . .                  | 209        |
| 8.3.3    | Learn-to-rank Models . . . . .                            | 212        |
| 8.3.4    | Refined Type-Sense Disambiguation and Modelling . . . . . | 213        |
| 8.3.5    | Refined Type Disambiguation . . . . .                     | 214        |
|          | <b>Bibliography</b>                                       | <b>215</b> |
| <b>A</b> | <b>Technical Background</b>                               | <b>229</b> |
| A.1      | Natural Language Processing . . . . .                     | 229        |
| A.1.1    | Word Tokenisation and Sentence Segmentation . . . . .     | 229        |
| A.1.2    | Part-of-Speech Tagging . . . . .                          | 230        |
| A.1.3    | Dependency Parsing . . . . .                              | 230        |
| A.1.4    | Word Embeddings . . . . .                                 | 230        |

|          |   |            |
|----------|---|------------|
| A.1.5    | The C-value/NC-value algorithm . . . . .  | 233        |
| A.2      | Machine Learning . . . . .  | 233        |
| A.2.1    | Support Vector Machines . . . . .   | 234        |
| A.2.2    | Neural Networks . . . . .   | 236        |
| A.2.2.1  | Cost Functions . . . . .  | 240        |
| A.2.2.2  | Activation Functions . . . . .  | 241        |
| A.2.2.3  | Optimisers . . . . .  | 243        |
| A.2.2.4  | Convolutional Neural Network Architecture . . . . .   | 243        |
| A.2.2.5  | Long Short-term Memory (LSTM) Architecture . . . . .  | 246        |
| A.3      | Significance Testing . . . . .  | 249        |
| <b>B</b> | <b>Test Collection Analysis Examples</b>  | <b>255</b> |
| B.1      | Example of the Application of my Process . . . . .  | 255        |
| B.2      | Full-scale Example Forms . . . . .  | 258        |
| <b>C</b> | <b>Instructions to Annotators for Gold-Standard Type List</b>                                     | <b>265</b> |
| <b>D</b> | <b>Supplementary Material and Results for Chapter 7</b>   | <b>269</b> |
| D.1      | Section 7.1 . . . . .   | 269        |
| D.1.1    | Comparative Results of TypeExp model . . . . .  | 269        |
| D.2      | Section 7.2 . . . . .   | 271        |
| D.2.1    | Subset and Statistics Definitions for Post-Hoc Analyses . . . . .                                 | 271        |
| D.2.2    | Subset $\mathbf{Q}_F$ . . . . .   | 272        |
| D.2.3    | Subset $\mathbf{Q}$ . . . . .   | 272        |
| D.2.4    | Subset $\mathbf{Q}_{FT}$ . . . . .  | 274        |
| D.2.5    | Section 7.2, Post-hoc Analyses . . . . .  | 276        |
| D.2.5.1  | Post-hoc Analysis 2 . . . . .   | 277        |
| D.3      | Section 7.3 . . . . .   | 277        |
| D.3.1    | Significance Tests between All Models based on Types2ExExp . . . . .                              | 277        |
| D.3.2    | Typed Retrieval Results on $\mathbf{Q}$ , $\mathbf{Q}_F$ and $\mathbf{Q}_{FT}$ (DevSet) . . . . . | 278        |
| D.3.2.1  | DevSet Results on $\mathbf{Q}_{FT}$ . . . . .   | 278        |
| D.3.2.2  | Results on $\mathbf{Q}$ . . . . .   | 279        |
| D.3.2.3  | Results on $\mathbf{Q}_F$ . . . . .   | 281        |
| D.3.3    | Post-Hoc Analyses in Section 7.3 . . . . .  | 283        |
| D.3.3.1  | Post-Hoc Analysis 1 . . . . .   | 284        |
| D.3.4    | Comparison to Textual Models from Section 7.1, DevSet Tuning . . . . .                            | 285        |
| D.3.5    | Summary Model Bar plots . . . . .   | 286        |





# Chapter 1

## Introduction

Sarah is a post-doctoral researcher in mathematics. For some time, she has been wondering about the functor from  $A$ -Algebras to sets,  $R \mapsto R \otimes M$ , and under which conditions it would be representable by an  $A$ -scheme, given that  $A$  is a commutative ring and  $M$  is an  $A$ -Module. After a lot of reading and some thought, Sarah came up with an answer: when  $A$  is a Noetherian ring and  $M$  is projective and finitely generated.

To arrive at this answer, Sarah had to absorb ideas in papers from algebraic geometry, commutative algebra and representable functors, and think through how ideas from these mathematical subjects come together to form a mathematically correct proof. The next step for Sarah is to make sure that her proof is novel and has not already been published in a journal.

Sarah's work would be made easier by an intelligent search engine to which she could ask many of the questions that arose during the construction of her proof. Such a search engine would have been able to find precisely those papers she needed while working out her proof, and it would also allow her to check that her novel result is indeed novel. The goal of this thesis is to explore mathematical IR models that could help Sarah in her quest for mathematical answers. As a mathematician, Sarah thinks, reasons, and communicates her work using concrete mathematical ideas (structures, objects and notions, such as algebra, module and Noetherian ring) and formulae. In his study of the mathematical discourse, Ganesalingam (2008) observed that the language of mathematics is different from normal text in two ways. First, written mathematics is more constrained, in terms of variation and linguistic phenomena, than general-purpose language. Second, the meaning of mathematical text is conveyed through the interaction of two modalities: the textual modality (flowing text) and the mathematical (or symbolic) modality (mathematical formulae). Ganesalingam also demonstrated that because of the condensed nature of information in the mathematical discourse, it can only be fully realised by unfolding the interactions between the two modalities. One such interaction that inspired my second hypothesis is the assignment of denotations (types) to elements of the symbolic context (variables in formulae). These observations led me to the two following hypotheses.

**Hypothesis 1.** Mathematical ideas, taking the form of multi-word technical terms in mathematical text, are broken up into their constituent words by current IR and MIR models. The implicit assumption these models make is that each constituent word is an orthogonal, independent source of information. I am curious to see, in the context of research-level mathematics, what happens to the efficiency of a Math Retrieval engine if technical terms that refer to mathematical ideas, which I will henceforth call *types*, are treated as atomic units of information.

**Hypothesis 2.** Math Retrieval might also profit from a linking of information from the textual modality with the symbolic modality. Specifically, I hypothesise that retrieval models that use types extracted from the text as denotations to symbols in formulae will have better retrieval efficiency than those that do not link text and formulae but treat them independently.

My investigation of these hypotheses starts with a test collection. In Chapter 3 I will present a novel process for constructing test collections for research-level mathematics retrieval evaluation and close the chapter by introducing the test collection I produced by applying my process.

In Chapter 4 I develop the idea of mathematical types further. At the start of the chapter I will define types in more detail and introduce four informal type inference rules. One rule is the basis for discovering types in text with the remaining three corresponding to critical components for realising typed retrieval (Chapter 6). I will then present algorithms for detecting and extracting types at scale and close the chapter with concrete retrieval models designed to investigate hypothesis 1.

Next, in Chapter 5 I address a prerequisite to investigating hypothesis 2: how to assign types from the textual modality as denotations to variables. I treat this as a machine learning task I call *variable typing* and in the same chapter I will introduce and evaluate machine learning models for the task.

In Chapter 6 I will describe how I realised retrieval models to investigate hypothesis 2. Specifically, I will describe how variable typing and my type inference rules from Chapter 4 can be used together to design retrieval models that link modalities. Towards the end of the chapter, in Section 6.4, I will introduce concrete retrieval models relevant to my second hypothesis.

I will compare the new models from Chapters 4 and 6 to current MIR systems in a large evaluation exercise in Chapter 7. My results show that treatment of types is overall beneficial for MIR. In particular, I found that one of the best MIR models I examined is textual and uses types found in the textual context of queries to expand them with similar types from an embedding space. Other successful models combine type-based textual retrieval with a novel form of tree matching with types that goes beyond pure syntactical similarity.

As a result of the work in this thesis, research mathematicians such as Sarah should be far better served with one of my new MIR models than with what was available when work on this thesis started. I will start this thesis with a review of that state of play.

# Chapter 2

## Background and Related Work

The primary goal of my thesis is to investigate methods for improving retrieval of research-level mathematics. In this chapter, I will introduce the necessary background in information retrieval (section 2.1) and in particular, review relevant advancements in mathematical information retrieval (section 2.2).

Section 2.3 is an in-depth overview of IR evaluation with discussions on evaluation measures, desirable properties of test collections and assumptions of retrieval experiments. In Section 2.4 we will look at methodologies for building test collections and identify important principles in test collection design. The material in both sections directly supports my test collection construction process (Chapter 3) and influenced the design of my experiments (Chapter 7).

An excursion into the linguistic and semantic description of mathematical text will follow in section 2.5. This is necessary because the approach I choose to follow, which relies on both textual and symbolic information from the mathematical text, is informed by the empirical and theoretical observations in this section. Finally, I will summarise the findings of this chapter and outline my approach for the rest of this thesis.

### 2.1 Overview of Information Retrieval

The process of retrieval starts with an *information need* – a user’s desire for information that relates to some topic. This information need is not observable from the outside and has to be expressed in a way that can be interpreted by the IR system, using the IR system’s input constraints, such as its input language. This concrete expression of the information need is referred to as the *query*. It is important to distinguish the initial information need from the query that represents it since the latter is a noisy expression of the former.

The user is looking for pieces of information that are *relevant* to the information need, i.e., answer it. In general, relevance is a problematic notion in IR evaluation (Katter, 1968; Burgin, 1992; Cooper, 1971; Saracevic, 1975; Schamber et al., 1990; Harter, 1996; Mizzaro, 1997), because relevance is known to be subjective (Schamber, 1994) and to differ between people and across time (Voorhees, 1998).

Given a query, the IR system is responsible for identifying pieces of information that are relevant to the searcher's request. It uses its stored internal representation of the documents<sup>1</sup> to do so. The set of all documents upon which retrieval is performed is called the *document collection*.

The retrieval operation is understood as a mapping from the query abstractions to the document (indexed) abstractions (Dubin, 2004). This mapping is at the core of an IR system and is reflected by the *retrieval model*, which is comprised of an indexing and a matching component (Dubin, 2004; Manning et al., 2008). At the highest conceptual level, the goal of a retrieval model is to represent documents, queries and their features in terms of an underlying abstraction. By this definition, the process of identifying relevant documents involves the computational construction of relationships between query and document abstractions.

Designing and implementing retrieval models involves a number of steps (Manning et al., 2008):

1. Construction of abstractions for each document in the collection.
2. Application of transformations on the document representations (such as *stop-word* filtering and *stemming*) and insertion in a suitable data-structure for efficient retrieval (e.g. an inverted index) and storage (e.g. a balanced tree). The first two steps are known as *indexing*.
3. Construction of a given query's representation in the underlying abstraction.
4. Application of transformations on the query model (such as *query expansion* if some information about relevance is available a priori).
5. *Matching* of the query model to the models of documents in the collection and/or application of the *ranking (or scoring) model* so that document models can be ordered by relevance to the query.

Modern IR is characterised by four classes of IR models: the Boolean retrieval model, heuristic models, classical probabilistic models, and models based on language modelling. Underpinning these models is the assumption that information (query and documents) can be represented by a *bag-of-words* – an unordered multi-set of words.

### **Boolean Model**

The Boolean retrieval model is the earliest successful IR model based on the bag-of-words abstraction (Taube and Wooster, 1958). In this model, documents are represented as bag-of-words multi-sets while queries take the form of nested Boolean expressions connected by Boolean operators (i.e. AND, OR and NOT) that have individual words (also known as terms) as their atomic operands. Queries are interpreted by a Boolean retrieval system in set-theoretic terms:

---

<sup>1</sup>Although an IR system can store information in many forms (e.g., images, scientific papers, books etc.), it is common to use the term “document” to refer to the smallest indexable unit of information.

the “meaning” of a Boolean expression corresponds to the set of documents in the index that satisfy it. Thus, relevance in Boolean retrieval is binary – a document is either a match or is assumed to be irrelevant.

The Boolean retrieval model can be effective under certain conditions: when all relevant documents must be retrieved (e.g., in a legal situation) and when relevance can be precisely defined using exact matching of query keywords. In practice, however, the Boolean retrieval model has been found to be too restrictive when using AND connectives and too broad when using OR operators (Lee and Fox, 1988). More importantly, the Boolean retrieval model is inflexible in scenarios where the information need is vague as it does not allow approximate matching through graded relevance.

### **Heuristic Models**

The *Vector Space Model* (VSM) (Salton et al., 1975) is the most well-known of the so-called heuristic IR models. It is an instance of the bag-of-words paradigm that assigns a score to each query-document pair that reflects the degree of relevance of that document to the query. Under the VSM, documents and queries are reduced into a smaller set of words, known as the *vocabulary*, by selecting  $n$  discriminating terms used to characterise them. The VSM abstraction then enables universal representation of queries and documents as vectors in an  $n$ -dimensional vector space, with the discriminating words in the vocabulary assuming the role of basis vectors (axes).

Thus, quantifying the correspondence/relevance between documents and query becomes straightforward: The relevance (or rank) of a document is defined to be the cosine of the angle between its vector and the vector of the query.

The VSM is often used in combination with the *TF-IDF* weighting scheme (Spärck-Jones, 1972), by which values are assigned to each component of the vectors. The weighting scheme is based on two statistics: The *term frequency* (TF), which is monotonically related to the number of times a vocabulary term appears in a document or query and the *inverse document frequency* (IDF), which is monotonically related to the number of documents that contain a term. In essence, the TF captures the intuition that the more frequently a query term occurs in a document, the more likely it is to be relevant to the query. IDF is used to determine how much weight to place on the discriminating ability of a term’s frequency in a document – the more documents a term appears in, the smaller its discriminating power. Therefore, assignments of component values based on the product of TF and IDF take into account the strength of relevance of the term in a way that is normalised by its overall discriminating ability.

Under the VSM with TF-IDF weighting scheme, a document  $d$  in the collection is represented by an  $n$ -dimensional vector

$$\vec{d} = \begin{bmatrix} \text{TF-IDF}_{1,d} \\ \vdots \\ \text{TF-IDF}_{n,d} \end{bmatrix}$$

Each component in the vector is the tf-idf weight of term  $t$  in document  $d$  in the collection:

$$\text{TF-IDF}_{1,d} \propto \text{TF}_{t,d} \times \text{IDF}_t.$$

Similarly, the query is represented by a vector  $\vec{q}$ :

$$\vec{q} = \begin{bmatrix} \text{TF-IDF}_{1,q} \\ \vdots \\ \text{TF-IDF}_{n,q} \end{bmatrix}$$

The length  $n$  of the vectors  $\vec{d}$  and  $\vec{q}$  is the aforementioned size of the vocabulary. The score (or degree of relevance) of document  $d$  to query  $q$  can be computed easily using the normalised dot product of the two vectors:

$$\text{score}_{d,q} = \frac{\vec{d} \cdot \vec{q}}{|\vec{d}| |\vec{q}|} \quad (2.1)$$

The score in Formula 2.1, also known as the *cosine similarity* score, allows documents to be ranked in order of relevance: the larger the cosine score of a document, the higher it is ranked.

### Classical Probabilistic Models

**BM25** Classical probabilistic models aim to estimate the probability of a particular document belonging to the “relevance” class of documents of a particular query. Proposed by Robertson et al. (1994), BM25 combines ideas from classical probabilistic models with ideas from the 2-Poisson model (Harter, 1975; Robertson et al., 1981; Robertson and Walker, 1994). The BM25 formula is a heuristic approximation of two Poisson distributions (i.e. a 2-Poisson model), one intended to model elite terms (terms that occur significantly frequently) and another to model non-elite terms (terms that occur by chance), conditioned on relevance (Harter, 1975; Robertson et al., 1981; Robertson and Walker, 1994; Robertson et al., 1994). Given a query  $q = q_1 \dots q_n$ , scoring with BM25 takes the following form:

$$\text{Score}(q, d) = \sum_{i=1}^n \text{IDF}(q_i) \times \frac{(k_1 + 1) \times \text{tf}_{q_i,d}}{\text{tf}_{q_i,d} + k_1 \times (1 - b + b \times \frac{|D|}{|D|_{\text{avg}}})}$$

**Models based on Language Modelling** Language modelling in IR is based on the *query likelihood model*, which aims to rank each document  $d$  by the probability that it has generated query text  $q$ :

$$P(d|q) = \frac{P(q, d)}{P(q)} = \frac{P(q|d)P(d)}{P(q)} \propto P(q|d)P(d)$$

Smoothing of the classical multinomial model is necessary to overcome problems related to maximum likelihood estimation, such as assignment of zero probabilities to unseen words (Zhai and Lafferty, 2001):

$$P(q|M_d) \propto \prod_{t \in V} P(t|M_d)^{tf_{t,q}}$$

where  $M_d$  is the multinomial language model for document  $d$ .

**Multinomial LM with Jelinek-Mercer smoothing ( $MLM_{jm}$ )** Jelinek-Mercer smoothing is a linear combination of the maximum likelihood estimation of a document model with that of a *background model*. A background model is a prior probability distribution of words, typically constructed from the document collection (Zhai and Lafferty, 2001):

$$P_\alpha(t|d) = (1 - \alpha)P_{ml}(t|d) + \alpha P(t|C)$$

**Multinomial LM with Dirichlet smoothing** Smoothing of the document model can also be performed using a Dirichlet distribution, which is the conjugate prior of the multinomial. The document language model in this case becomes:

$$P_\mu(t|d) = \frac{tf_{t,d} + \mu P(w|C)}{\sum_t tf_{t,d} + \mu}$$

**SPUD** is a state of the art unigram language modelling IR model (Cummins et al., 2015). It models documents as draws from a multivariate Polya distribution. Smoothing of the document model is performed using a linear combination of the unsmoothed document and background models and is controlled through a smoothing parameter  $\omega$ . The SPUD ranking method estimates the probability that a query is generated from the expected multinomial drawn from each document model:

$$SPUD(q, d) = \sum_{t \in q} \left( \log \left( \frac{(1 - \omega)|\vec{d}| \frac{tf_{t,d}}{|\vec{d}|} + \omega m_c \frac{df_t}{\sum_j |\vec{d}_j|}}{(1 - \omega)|\vec{d}| + \omega m_c} \right) \right) tf_{t,q}$$

In the corresponding bigram LM the query likelihood is estimated by

$$P(q|M_d) \approx \prod_{i=1}^n P(q_i|q_{i-1}, M_d)$$

Language models can be extended with query expansion methods, which introduce new terms

in queries to aid retrieval. The RM3 method assumes that documents and queries are generated by the same relevance model. RM3 tries to avoid generating topically unrelated expansion terms from the background model by linearly combining the smoothed document models using a prescribed number of top ranking documents (Abdul-jaleel et al., 2004; Lv and Zhai, 2009). The PRM2 relevance model (Lv and Zhai, 2011) uses proximity information in the feedback documents to expand documents.

## 2.2 Mathematical Information Retrieval

Mathematical Information Retrieval (MIR) is a sub-discipline of IR that addresses the task of retrieving mathematical information. In their survey of retrieval of mathematical knowledge, Guidi and Coen (2015) conclude that the few MIR systems under active development demonstrate low precision and recall scores compared to other domains. I have split the models I will describe into early models (those that represent formulae as bags of terms), tree-based models, the MCAT system, and TANGENT. Before surveying these MIR models, I will first describe the format in which mathematical expressions are commonly stored.

### 2.2.1 Representation of Mathematical Expressions

The meaning of formulae is represented by two components: the overall structure of a formula and the lexical tokens (symbols) it encapsulates (Nghiem et al., 2014). Mathematical formulae are hierarchical in nature and are constructed by nesting expressions of arbitrary complexity (Kamali and Tompa, 2009; Nghiem et al., 2014). This tree-like structure encodes mathematical relationships (e.g., mappings, function application, equality, set membership and operator/operand relationships) between objects abbreviated by the lexical symbols in the expression. The question is how to represent this structure.

Most scientific documents, such as those published on ArXiv, represent mathematical expressions as  $\text{\LaTeX}$  math environment command blocks. However, the  $\text{\LaTeX}$  language was designed as a typesetting language and hence does not allow the semantic annotation of mathematical expressions. In response to this, the MathML standard (an application of the XML markup language) was proposed to standardise the representation and distribution of mathematical content.

The MathML standard defines two subsets that complement each other: *Presentation* and *Content* MathML. Presentation MathML describes the expression as a visual object, i.e., how mathematical notation is to be displayed in documents and screens, while Content MathML is used to describe semantics – which symbols are operators, as opposed to operands, and in which order the operators are applied.

Figure 2.1 illustrates the difference by showing the example expression  $\left(\frac{p-2}{p-1}\right)^{p-1}$  in both formats. The main component of the expression’s Presentation MathML is seen as a `<mfrac>`

| Presentation                        | Content                             |
|-------------------------------------|-------------------------------------|
| <code>&lt;msup&gt;</code>           | <code>&lt;apply&gt;</code>          |
| <code>&lt;mfenced&gt;</code>        | <code>&lt;exp/&gt;</code>           |
| <code>&lt;mfrac&gt;</code>          | <code>&lt;apply&gt;</code>          |
| <code>&lt;mrow&gt;</code>           | <code>&lt;divide/&gt;</code>        |
| <code>&lt;mi&gt;p&lt;/mi&gt;</code> | <code>&lt;apply&gt;</code>          |
| <code>&lt;mo&gt;-&lt;/mo&gt;</code> | <code>&lt;minus/&gt;</code>         |
| <code>&lt;mn&gt;2&lt;/mn&gt;</code> | <code>&lt;ci&gt;p&lt;/ci&gt;</code> |
| <code>&lt;/mrow&gt;</code>          | <code>&lt;cn&gt;2&lt;/cn&gt;</code> |
| <code>&lt;mrow&gt;</code>           | <code>&lt;/apply&gt;</code>         |
| <code>&lt;mi&gt;p&lt;/mi&gt;</code> | <code>&lt;apply&gt;</code>          |
| <code>&lt;mo&gt;-&lt;/mo&gt;</code> | <code>&lt;minus/&gt;</code>         |
| <code>&lt;mn&gt;1&lt;/mn&gt;</code> | <code>&lt;ci&gt;p&lt;/ci&gt;</code> |
| <code>&lt;/mrow&gt;</code>          | <code>&lt;cn&gt;1&lt;/cn&gt;</code> |
| <code>&lt;/mfrac&gt;</code>         | <code>&lt;/apply&gt;</code>         |
| <code>&lt;/mfenced&gt;</code>       | <code>&lt;/apply&gt;</code>         |
| <code>&lt;mrow&gt;</code>           | <code>&lt;apply&gt;</code>          |
| <code>&lt;mi&gt;p&lt;/mi&gt;</code> | <code>&lt;minus/&gt;</code>         |
| <code>&lt;mo&gt;-&lt;/mo&gt;</code> | <code>&lt;ci&gt;p&lt;/ci&gt;</code> |
| <code>&lt;mn&gt;1&lt;/mn&gt;</code> | <code>&lt;cn&gt;1&lt;/cn&gt;</code> |
| <code>&lt;/mrow&gt;</code>          | <code>&lt;/apply&gt;</code>         |
| <code>&lt;/msup&gt;</code>          | <code>&lt;/apply&gt;</code>         |

Figure 2.1: Presentation MathML and Content MathML for Expression  $\left(\frac{p-2}{p-1}\right)^{p-1}$  (adapted from Aizawa et al., 2013).

tag with two arguments – one for the numerator and one for the denominator. The `<mrow>` tag expresses which symbols are to be displayed on the same typographical baseline (here: the numerator and denominator of the fraction respectively). Simple identifiers are expressed by `<mi>`, operators by `<mo>` and numbers by `<mn>`. The exponentiation is expressed typographically as a `<msup>` tag (for superscripted expressions), which takes as its argument the base and the script (here:  $p - 1$ ). Presentation MathML handles subscripts, tables and matrices in a similar way, by treating them as scripted sub-expressions in their respective positions.

Content MathML in contrast expresses the mathematical semantics directly, by specifying that the main operation we see is an exponentiation operator acting on the two large sub-expressions of a division, each of which is the outcome of a subtraction. The `<apply>` tag is used to specify the order in which operators are applied, and which operands they act on.

Much of today’s practical formula processing in MIR systems relies on Presentation MathML, for instance Schellenberg et al. (2012) (which uses SLTs, cf. below) and TANGENT (which uses triples), all of which were derived from Presentation MathML. I will also use Presentation MathML in this work.

There is consensus in the MIR community that parallel Presentation/Content MathML markup is beneficial to MIR retrieval performance (Guidi and Coen, 2015; Kristianto et al., 2014a;

Ruzicka et al., 2014; Lísška et al., 2013), but Content MathML remains to be of limited availability. At the time of writing, no large document collections with natively annotated Content MathML (Kristianto et al., 2014a; Kamali and Tompa, 2010; Guidi and Coen, 2015) exist, due to the fact that the automatic conversion of Presentation MathML to Content MathML is computationally difficult and thus unreliable (Guidi and Coen, 2015; Nghiem et al., 2014). There are however some efforts to overcome this problem, for instance by enriching Presentation MathML formulae with surrounding text (Quoc et al., 2010; Pinto et al., 2014; Kristianto et al., 2014a; Kristianto et al., 2014b; Guidi and Coen, 2015). Content MathML has other disadvantages aside from its practical unavailability: there is no dictionary of terms agreed on by all authors of Content MathML, and it would be nontrivial to agree on such a dictionary, given that the set of symbols and concepts across mathematical disciplines is open (Kamali and Tompa, 2009; Ganesalingam, 2008). Others have argued that the non-availability of some information encapsulated in the layout of symbols can result in lower precision in MIR (Guidi and Coen, 2015; Pinto et al., 2014). Kamali and Tompa (2009) also note that for most use-cases of MathML, such as distribution of documents on the web, Content MathML amounts to “overkill” as it is not necessary to specify the meaning of each symbol.

Content MathML therefore is mainly used outside MIR for the exchange of information between knowledge management systems (Guidi and Coen, 2015), although there is at least one system that uses Content MathML for a task closely related to MIR (namely the MCAT system by Kristianto et al. (2014a)).

*Symbol Layout Trees* (SLTs) are representations for formulae widely used in MIR. SLTs are tree structures that represent mathematical expressions in terms of the 2-dimensional layout of their symbols. Nodes in SLTs can either be terminals, representing concrete symbols, or non-terminals. Non-terminals are internal SLT nodes whose sub-tree represents structurally rich mathematical expressions (e.g., matrices or fractions) that are sub-terms to the main expression represented by the SLT itself. Edges in SLTs are directed and labelled with the spatial relationship between the nodes they connect.

As an example, consider the SLT for the formula  $\frac{x^2+y}{\sqrt{z}}$  shown in Figure 2.2. The non-terminal node `Frac`, with two sub-trees in the `ABOVE` and `BELOW` direction, models the overall layout of the formula: a fraction with a numerator term at the top and a denominator term at the bottom position. Similarly, the `Sqrt` non-terminal models the layout of a square rooted term: the argument to a square root appears inside the square root operator. The symbols  $x$ ,  $+$ ,  $y$ ,  $z$  and the literal 2 are the terminal nodes of the SLT. The overall layout of the sub-expression  $x^2 + y$  of the formula is represented by a sequence of `ADJ` (adjacent) edges encoding the fact that the connected nodes are placed along the same (typographical) line.

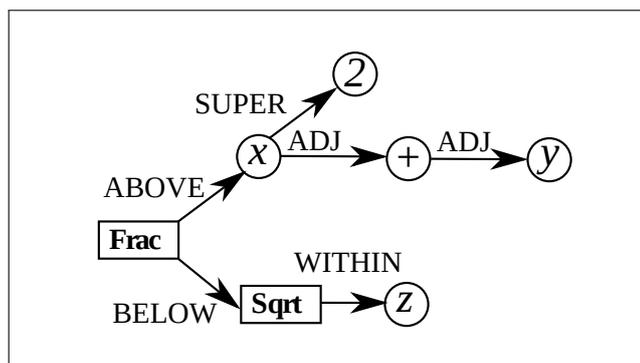


Figure 2.2: SLT for the formula  $\frac{x^2+y}{\sqrt{z}}$  (adapted from from Pattaniyil and Zanibbi (2014)).

There are two decisions affecting the representational fidelity of SLTs: the kinds of edges and the vocabulary of terminal and non-terminal nodes corresponding to formula constituents. Pattaniyil and Zanibbi (2014) make reference to at least five kinds of SLT edges: ADJ (adjacent), ABOVE, BELOW, SUPERSCRIPT and WITHIN and to non-terminal SLT nodes for matrices and fractions in their examples but do not give an exhaustive list for types of edges or non-terminals. Schellenberg et al. (2012) constructed SLTs from Presentation MathML with spatial relationships in the ABOVE, SUPERSCRIPT, BELOW, SUBSCRIPT, RIGHT directions and a spatial edge for function arguments (in the RIGHT direction).

I adopt SLTs to represent formulae but with some extensions which I will describe in Section 6.1.1 and motivate further in Chapter 6.

### 2.2.2 Non-structural MIR models

Early work in MIR investigated the effectiveness of bag-of-words and visual methods in formula retrieval. Zanibbi and Yuan (2011) investigated two formula retrieval methods. The first translated expressions from  $\text{\LaTeX}$  into strings which were then indexed in a normal bag-of-word manner using the Lucene system. For the translation they used a dictionary of keyword substitutions, which included keywords for spatial relationships. For example, the expression  $\alpha + \beta$  translates to “alpha plus beta” and  $x^y$  translates to “x superscript y”. The second method is image-based: contour, density, aspect ratio and relative position features are extracted from images of mathematical expressions using information on connected components. Formula retrieval in this model is based on minimum cost correspondence between the features of the query and document formulae, implemented using a greedy algorithm. The evaluation<sup>2</sup> of this system used 10 randomly selected mathematical expressions as queries and 50 documents containing 26,737 expressions. Relevance judgements were manually produced. The keyword and image-based retrieval methods achieved weighted precision-at-k ( $k = 20$ ) scores of 0.72 and 0.24 respectively. 2 out of the 10 queries in image-based retrieval yielded a precision-at-k score of 0. A combined keyword and image-based retrieval approach yielded a precision-at-k ( $k = 20$ )

<sup>2</sup>All concepts related to evaluation will be introduced later in section 2.3.

score of 0.65.

Larson et al. (2013) performed formula search by combining traditional keyword-based Boolean retrieval with bitmap indexing of operators in MathML trees. A query is produced as a conjunction of three bags of objects:

- the keywords in the formula search topic
- a string representation of the associated formula produced by concatenating the text elements of the MathML tree with specific string translations of spatial relationships (e.g., “superscript” for the `<msup>` tag); and
- an incidence vector of operators present in the formula’s MathML tree.

Retrieval is performed by using part (a) of the query to obtain 100 preliminary document results (the “preliminary search” step). Parts (b) and (c) are then used to match formulae in the document with the formula in the query.

This approach achieved poor performance with MAPs as low as 0.0018. Document rankings produced during the preliminary search step (Larson et al., 2013; Aizawa et al., 2013) were also found to be poor. An attempt to improve the preliminary search step with hand-crafted reformulations of component (a) matched against document titles did not improve results either. Larson et al. (2013) conclude that:

*“math search is both qualitatively and quantitatively different from ordinary search and will require significant development of new ideas and approaches to be able to achieve the goal of mathematical information retrieval”.*

Guidi and Coen (2015) highlight that early work in MIR often didn’t perform comparisons to existing algorithms in the literature or used benchmarks that were unclear or could not be reproduced. They also mention that the main problem in MIR evaluation is the absence of real world, interesting queries for evaluation, given that MIR systems are known to be highly sensitive to the kind of queries used. The authors identify three problems with MIR evaluation: (a) automatic evaluation in MIR is particularly hard while manual evaluation is limited to a small number of queries and runs, (b) formulating good query sets is complex because users with different mathematical backgrounds and motivation are likely to issue different queries, and (c) queries might be formulated at the wrong level of abstraction, make use of non standard notation or even have errors in formula encoding.

### **2.2.3 Tree-based MIR models**

One one hand, reducing formula retrieval into full-text, bag-of-words search eliminates information about the structure of formulae: the structural relationships between symbols is hard to preserve under a bag-of-words model (Guidi and Coen, 2015).

On the other hand, exact tree matching methods are expensive and too strict in that they penalise synonymous expressions (Kamali and Tompa, 2009; Kamali and Tompa, 2010; Guidi and Coen, 2015): for instance, Kamali and Tompa (2013) proposed tree edit distance as a measure of formula similarity, but this approach is expensive to implement and requires substantial algorithmic optimisations to be practical (Guidi and Coen, 2015). For these reasons, MIR researchers have adopted a heuristic approach to tree similarity (Guidi and Coen, 2015). This compromise comes at a cost: most heuristic methods in the literature are incomparable to each other, as Guidi and Coen (2015) note.

The tree matching algorithm for formula retrieval by Kamali and Tompa (2009) considers the Presentation MathML trees of two formulae  $E_1$  and  $E_2$  a perfect match if they have the same structure and the nodes at corresponding points on the tree have the same labels. The notion of similarity is based on the weight function  $w$ , which assigns a weight of 1 to any vertex of  $E$ . The weight of a sub-tree  $w(E)$  is the sum of all its vertex weights, effectively making the weight of the sub-tree equal to the number of nodes it contains.

Similarity between  $E_1$  and  $E_2$  is then defined as a ratio of the weights of the sub-trees common to both  $E_1$  and  $E_2$ , normalised by the weights of  $E_1$  and  $E_2$ :

$$\text{sim}(E_1, E_2) = \frac{w(E_1 \cap E_2)}{w(E_1) + w(E_2)},$$

where  $w(E)$  is the weight of the tree representing sub-expression  $E$  and  $E_1 \cap E_2$  is the common sub-tree of  $E_1$  and  $E_2$  that matches exactly.

Kamali and Tompa also identified four patterns of structural similarity between two mathematical expressions  $E_1$  and  $E_2$ :

- a) identity if  $E_1$  and  $E_2$  are exactly the same in terms of structure and symbols
- b)  $\alpha$ -equivalence if the structure is identical and symbols are mappable. In particular, two formulae  $E_1$  and  $E_2$  are  $\alpha$ -equivalent if they have the same structure and there is a set of substitutions that transform the atomic terms (variables) of one formula into the other. For example,  $x^2 + y$  and  $a^2 + b$  are  $\alpha$ -equivalent because they share the same structure and there exists a set of substitutions,  $x \rightarrow a$  and  $y \rightarrow b$ , that maps the symbols from one formula to the other without conflict. However, the expressions  $y^2 + y$  and  $y^2 + x$  are not alpha equivalent because the substitution set contains conflicting substitutions:  $y \rightarrow y$  conflicts with  $y \rightarrow x$  since  $y$  cannot be mapped onto two different variables<sup>3</sup>.
- c) mathematical equivalence, if  $E_1$  and  $E_2$  can be mathematically proven to be equivalent, and
- d)  $n$ -similarity. Two formulae  $E_1$  and  $E_2$  are  $n$ -similar if  $\text{sim}(E_1, E_2) \geq n$ , where  $n$  is a parametric threshold on the similarity between two expressions.

---

<sup>3</sup>The same restriction applies in the other direction too, as no two variables can be mapped onto the same variable.

Unfortunately, Kamali and Tompa did not perform an experimental evaluation of the proposed methods.

Another idea used in some MIR experiments is substitution tree indexing (Graf, 1994). Substitution tree indexing can be used for semantic indexing and exact matching of formulae up to  $\alpha$ -equivalence. Kohlhase and Sucan (2006) adapted substitution tree indexing to index Content MathML in a system called MathWebSearch. A substitution tree index for Content MathML is a tree where each edge is a substitution and each node is formula. An internal node is an abstraction of its children but a specialisation of its parent. Common terms between a node and its children become substitutions encoded by the edges and are replaced by placeholders in the node. Concrete formulae (i.e., those without any placeholders) can be retrieved by following a path from the root down to specific leaves of the substitution tree index. For example, consider the following five formulae (Kohlhase and Sucan, 2006):

$$\begin{aligned} &h(f(z, a, z)), \\ &\quad x, \\ &g(f(z, y, a)), \\ &g(f(7, z, a)), \\ &g(f(7, z, f)) \end{aligned}$$

These formulae share structure – the three instances of  $g$  have different, but similar instances of  $f$  as their argument. The substitution tree index for these formulae encodes their shared structure as shown in Figure 2.3. The root of the substitution tree is a placeholder ( $@0$ ) that can be substituted with any formula. The formula  $h(f(z, a, z))$  and the variable  $x$  do not share any structure with the other formulae (the variable  $x$  is a different term, while the other formulae have  $g$  instead of  $h$ ) and are stored as leaves in the tree via the substitutions  $@0 \mapsto h(f(z, a, z))$  and  $@0 \mapsto x$  respectively. The remaining formulae share structure but have different instances of the sub-term  $f$ . The node  $g(f(@1, @2, a))$ , like all internal nodes, is a specialisation of its parent; it is produced by replacing the placeholder  $@3$  in its parent with the term (variable)  $a$ . It is also a generalisation of its children – each of its two children specialises the placeholders  $@1$  and  $@2$  with different terms. MathWebSearch is not formally evaluated, and has been criticised by Guidi and Coen (2015) for not providing a ranking.

Zhang and Youssef (2014) criticised MathWebSearch’s inability to provide relevance ranking or similarity search, whereas Guidi and Coen (2015) state that the approach is sophisticated and has the potential for high precision, but has limited applicability in MIR due to poor recall.

Substitution tree indexing was also adapted by Schellenberg et al. (2012) to index and match formulae converted to SLTs with five kinds of edges (ABOVE, SUPERScript, BELOW,

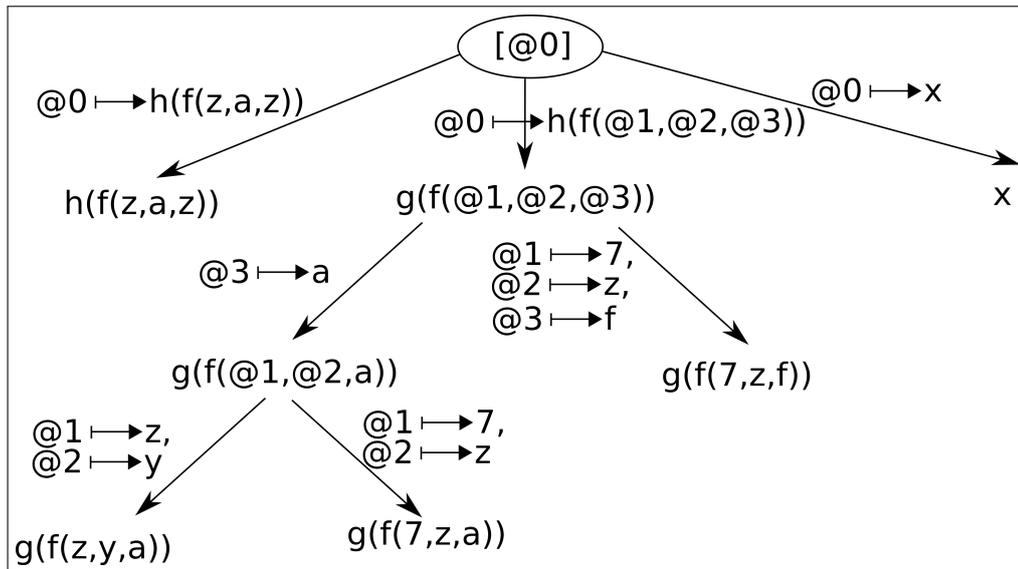


Figure 2.3: Example of a substitution tree index (adapted from Kohlhase and Sucan (2006)).

SUBSCRIPT, RIGHT) from Presentation MathML. The authors evaluated their implementation on the same 10 expression as Zanibbi and Yuan (2011) and reported similar performance to that observed by Zanibbi and Yuan (2011).

Zhang and Youssef (2014) proposed a VSM-based similarity metric based on five features extracted from strict Content MathML. Their similarity function scores two MathML trees  $T_1$  and  $T_2$ , representing the query and document formulae respectively, through simultaneous traversal. During this process, the following features are extracted from the trees:

1. Taxonomic distance of functions. Similarity between two terms is defined in terms of taxonomic similarity, using keywords extracted OpenMath CDs, and is maximised if they belong to the same OpenMath CD.
2. Data type hierarchical level. This feature captures similarity of Content MathML trees based on the types of MathML nodes in formulae. Nodes can be constants, variables or functions, and some data types contribute more significance to overall similarity than others. For example, nodes that represent higher-level constructs, such as functions, are more significant indicators of similarity than lower-level nodes such as variables.
3. Matching depth. The depth of the node at which a query expression  $q$  is matched on the MathML tree of a document formula  $d$  is used as an indicator of how similar  $q$  is to  $d$ . Matching a query formula further down the MathML tree of the document formula suggests that the former is only a small, locally important component of the latter. In contrast, matching a query formula higher-up in the MathML tree of a document formula suggests that there is more comprehensive similarity between the two expressions.
4. Query Coverage. This feature records the proportion of the query formula (e.g., in terms of

number of nodes) that is covered by a document formula.

5. Formula vs. Expression. Expressions that have relational operators (e.g.,  $\geq$ ,  $=$  etc.) as the root of their Content MathML tree are ranked higher than expressions that do not.

Zhang and Youssef compared their method to the equation-based math search system that is part of the Digital Library of Mathematical Functions (DLMF) project<sup>4</sup> (Miller and Youssef, 2003; Youssef, 2007) using 40 mathematical expressions from the DLMF that were hand-crafted into strict Content MathML specifically for their evaluation. This makes direct comparison of their method to other methods proposed in the literature infeasible.

#### 2.2.4 The MCAT model

Yokoi and Aizawa (2009) proposed a similarity metric based on subpath sets, computed over Content MathML using the Jaccard coefficient. The subpath set of a MathML tree is defined to be the set of all subpaths between any node on the tree to every leaf.

To illustrate this definition, the subpath set for the tree in Figure 2.4 contains 14 paths, namely

$$\begin{aligned} & \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \\ & \{a, b\}, \{a, c\}, \{b, d\}, \{b, e\}, \{c, f\}, \\ & \{a, b, d\}, \{a, b, e\}, \{a, c, f\}\} \end{aligned}$$

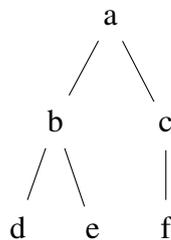


Figure 2.4: Example MathML tree with nodes a, b, c, d, e and f. Adapted from Yokoi and Aizawa (2009).

Similarity between two MathML trees  $T_1$  and  $T_2$  is measured as the overlap of their subpath sets using the Jaccard coefficient (Yokoi and Aizawa, 2009):

$$\text{sim}(T_1, T_2) = \frac{|\text{subpathset}(T_1) \cap \text{subpathset}(T_2)|}{|\text{subpathset}(T_1) \cup \text{subpathset}(T_2)|}$$

Yokoi and Aizawa performed a preliminary qualitative evaluation of their similarity method using 155,607 mathematical expressions. The authors concluded that the subpath set similarity

<sup>4</sup><http://dlmf.nist.gov/>

captured structural similarity between expressions well and worked best with Content, rather than Presentation MathML.

The similarity method proposed by Yokoi and Aizawa (2009) was adapted by the MCAT system (Kristianto et al., 2014a; Kristianto et al., 2016; Kristianto et al., 2014b). The MCAT system supports indexing and retrieval of both text and formulae using Apache Solr, a key-value database based on Lucene. Mathematical expressions are indexed from Presentation MathML using three path sets stored as strings: (a) vertical subpath sets in the order they appear on the MathML tree, (b) unordered vertical subpaths sets and (c) a “sisters” set encoding sibling nodes at each level. In addition, the MCAT system stores two fields for each indexed expression. One field is used to store keywords taken from a small window surrounding the target formula and another to store an automatically extracted description. The intention behind the aforementioned fields is to capture different methods of encoding the meaning of the representation.

Descriptions are extracted using an SVM (Kristianto et al., 2012). This leaves open the possibility that for some formulae no description can be detected (whereas window-based extraction methods always results in keywords). The MCAT system infers such missing descriptions from a dependency graph (Kristianto et al., 2014b; Kristianto et al., 2016), which connects every mathematical expression to the descriptions of its sub-expressions as shown in Figure 2.5. A description for an expression can be generated by computing the set union (over sets of keywords) of the descriptions of its sub-expressions. Scoring in the MCAT system is a linear combination of the content (window and description text) and structural features of the query ( $f_q$ ) and document ( $f_x$ ) formulae:

$$score(f_x, f_q) = \text{contSim}(f_x, f_q) \times \alpha + \text{structSim}(f_x, f_q) \times (1 - \alpha),$$

where `contSim` and `structSim` are the content and structure similarity functions implemented using the Apache Solr (Lucene) VSM. The parameter  $\alpha$  is set to 0.3 in order to give more weight to structural similarity.

The system was evaluated in the framework of the math search subtask of NTCIR-10 and subsequent NTCIRs. It scored between  $\text{MAP} = 0.162$  and  $\text{MAP} = 0.297$  on this task.

### 2.2.5 Tangent

Tangent (Pattaniyil and Zanibbi, 2014) is a state-of-the-art MIR system that, like MCAT, supports retrieval of both textual and symbolic contexts. Tangent’s indexing method can be applied at scale as it only approximates the structure of mathematical expressions.

Tangent represents formulae using SLTs created from Presentation MathML. Given a mathematical formula, the indexing algorithm generates symbol pair tuples in a depth-first manner starting from the root node of the SLT. Symbol pair tuples record parent/child relationships between SLT nodes, the distance (number of edges) and vertical offset between them. At each step in the depth-first traversal, the index is updated to record one tuple for every node along the

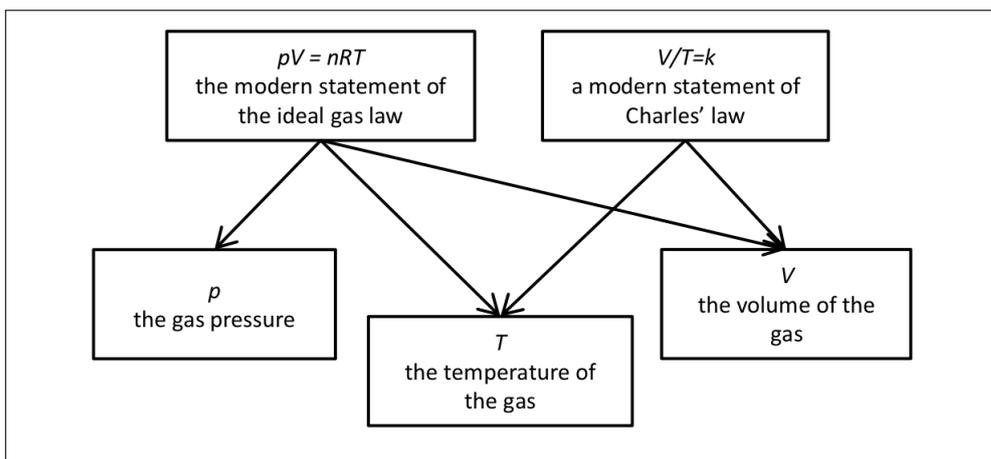


Figure 2.5: Search terms of the top-level formulae in MCAT expanded by sub-expression descriptions (from Kristianto et al., 2014).

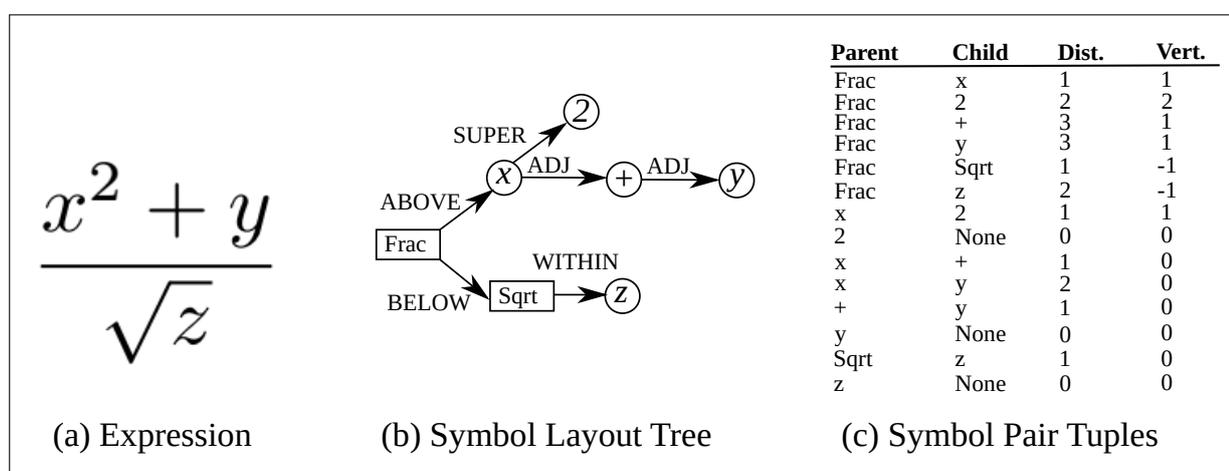


Figure 2.6: (a) Example formula, (b) Symbol layout tree, (c) Symbol pair tuple index for the formula. Figure adapted from Pattaniyil and Zanibbi (2014).

path to the root.

In order to illustrate, consider the Tangent formula indexing example presented in Figure 2.6. The expression  $\frac{x^2+y}{\sqrt{z}}$  in Figure 2.6(a) is represented by the SLT in figure 2.6(b). Traversal begins from the node of the SLT, `Frac`, but a tuple is emitted only when the child node  $x$  is reached. At this point, traversal has moved vertically one step (in the “ABOVE” direction) with a total traversal distance of 1. This information is encoded in the first row of Figure 2.6(c). Subsequently, traversal moves to node “2” by following a superscript edge on the SLT. One tuple is then generated to reflect the relationship between nodes “2” and “x” (row 7) and another to reflect the relationship between nodes “2” and the root `Frac` (row 2). This process is repeated until all nodes on the SLT are visited.

Formula indexing in Tangent, although conceptually similar to that of the MCAT system, is a heuristic approximation of the structure of indexed formulae. MCAT stores complete subpath

sets whereas Tangent breaks up the paths into pairs and stores positional offsets as supplemental structural features. Thus, Tangent relaxes the strictness of tree or subpath matching, making it potentially more flexible with respect to formula polysemy and synonymy.

Tangent scoring proceeds as follows. For each query formula, the symbol pair tuples are generated and matched exactly to those in the document index. Let  $C$  denote the set of matched index formulae and  $|s|$  the number of symbol pairs in any given expression  $s$  in  $C$ . For each  $s$  in  $C$ , recall (R)<sup>5</sup> is said to be  $\frac{|C|}{|Q|}$ , where  $|C|$  and  $|Q|$  are the numbers of tuples in  $C$  and the query formula  $Q$  respectively, and precision (P) is  $\frac{|C|}{|s|}$ . Candidate  $s$  is assigned the  $F$  score of these precision and recall values. The mathematical context score for a given document  $d$  and query with formulae  $e_1, \dots, e_n$  is

$$m(d, e_1, \dots, e_n) = \sum_{j=1}^n \frac{|e_j| \cdot t1(d, e_j)}{\sum_{i=1}^n |e_i|}$$

where  $|e_j|$  represents the number of tuples in expression  $e_j$  and  $t1(d, e_j)$  represents the top F-score for expression  $e_i$  in document  $d$ . The final score for document  $d$  is a linear combination of the math context score above and its Lucene text score (L(d)):

$$\alpha L(d) + (1 - \alpha)m(d, e_1, \dots, e_n)$$

Tangent indexing and scoring also supports matrices and wildcard formula queries<sup>6</sup>.

This concludes my review of important developments in MIR. I will now turn to a description of the language in mathematical articles, before laying out the approach chosen in this thesis.

## 2.3 Information Retrieval Evaluation

Evaluation in IR provides an assessment of how well a system satisfies the information needs of its users (Voorhees, 2002). The ability of an IR system to retrieve documents that are relevant to the searcher's information needs is referred to as *effectiveness*. Quantifying the effectiveness of an IR system often involves determining the relevance of retrieved documents relative to the searcher's information need.

Laboratory testing has been the prevailing experimental IR model since its introduction in the Cranfield II experiments (Cleverdon, 1967). Laboratory tests are necessary abstractions of the overall IR situation that give the experimenter full control over all variables. Not all aspects of the real world can be modelled and it is, therefore, the experimenter's responsibility to identify the most relevant aspects for modelling.

<sup>5</sup>Precision and recall will be formally defined in section 2.3.

<sup>6</sup>These features are not discussed here because they are not relevant in the framework of my own experiments in chapter 7, in which I will use Tangent as a baseline.

Central to the design and execution of laboratory experiments is the concept of the *information retrieval task*. An IR task is an abstraction of a specific retrieval problem to be solved. As an abstraction to a particular retrieval problem, IR tasks are usually accompanied by (a) a description of the problem that is meaningful to the IR research community and (b) supporting data and evaluation methods and measures necessary for comparative evaluation of systems that attempt to solve it.

The standard task in information retrieval is *ad-hoc retrieval*. An ad-hoc task is designed to simulate the scenario whereby one or more users are looking to satisfy information needs over a large collection of documents.

Examples of retrieval tasks other than ad-hoc retrieval and MathIR include *web search* (Hawking and Craswell, 2004) (retrieval over a large, dynamic and interconnected collection of heterogeneous documents) and *patent search* (Iwayama et al., 2003; Graf and Azzopardi, 2008), retrieval of highly technical legal patents that conform to a specific, predefined structure.

The experimental objectives of my thesis are abstracted into a special instance of the ad-hoc task: ad-hoc retrieval for research-level mathematical information needs. As such, my evaluation data is designed to enable IR experimenters to develop and comparatively evaluate retrieval models that take advantage of the intricacies associated with research-level mathematics. The intricacies of this task include (i) difficult, research-level mathematical problems, (ii) information needs expressed in the natural language of mathematics that are rich in formulae and (iii) documents that are large, research-level bodies of mathematical work.

### 2.3.1 The Test Collection Paradigm

Evaluation through laboratory tests in IR is carried out using a resource known as a *test collection*. A test collection consists of three components (Manning et al., 2008):

1. A collection of documents referred to as the *document collection* (or corpus).
2. A set of queries derived from abstractions of information needs (also referred to as topics).
3. A set of *relevance judgements* identifying each document in the document collection as relevant or nonrelevant for each query.

An IR system being evaluated with a test collection must first index the documents in the document collection. Then, the queries in the test collection are submitted to the system in sequence, with each query giving rise to a *results set* – a list of documents ranked in order of relevance. Finally, the results sets of all submitted queries are concatenated into a set known as a *run*, which is used to compute retrieval efficiency measures that reflect the performance of the IR system on the test collection.

Differences in retrieval effectiveness between systems (or different versions of the same system) can be identified by comparing runs of the systems obtained on the same test collection.

In fact, test collections have been widely adopted for comparative evaluation of retrieval systems (Salton, 1971; Spärck-Jones, 1981; Spärck-Jones and Willett, 1997; Harman, 1993; Voorhees and Harman, 2005). This wide adoption has prompted the IR research community to investigate the reliability of comparing IR system performance using test collections.

The IR community has identified the following factors as contributing to the reliability of comparative evaluation in IR (Buckley and Voorhees, 2000; Voorhees, 2002; Buckley and Voorhees, 2004; Voorhees and Harman, 2005):

- the suitability, quality and size of the chosen test collection (e.g., in terms of the number of topics and relevance judgements per topic);
- the methodology used to construct it; and
- the reliability of the performance measures used to assess performance efficiency.

Three properties of test collections influence the likelihood of performance gains obtained during simulation tests transferring to the real world.

**Property 1: *Realisticness*.** A test collection should be an accurate representation of the real-life operational setting it intends to simulate (Spärck-Jones and van Rijsbergen, 1976). The degree to which a test collection is a realistic simulation of its operational setting depends on the underlying document and query sets. According to Voorhees (2002), the document collection must be a “sample of the kind of texts that will be encountered in the operational setting of interest”. For example, the document set in a test collection intended to evaluate retrieval models for research level mathematics should be composed of Mathematics research papers, rather than entry-level mathematics text books. Furthermore, the queries attached to a realistic test collection should approximate real use cases and be accurate representations of the kinds of information needs that real users will likely try to satisfy using the IR systems being evaluated.

**Property 2: *Diversity*.** A test collection must be large and diverse enough to capture the variability of documents and users encountered in the real world (Voorhees, 2002; Spärck-Jones and van Rijsbergen, 1976). The document set must reflect, amongst other things, the diversity of the subject matter, variations in document length, word choice and writing style and the many different ways the same concept can be expressed in natural language. Similarly, the query set must be large enough to model user diversity, i.e., covering a wide range of topics and derived queries and diverse ways in which users of the evaluated retrieval systems are expected to express their information needs. As we will see in subsequent sections, the number of queries in a test collection also affects the reliability of the conclusions obtained from IR experiments.

**Property 3: *Reusability*.** A test collection is reusable if it can be used in different experimental configurations within the operational setting being simulated, i.e., if the same resource can be used to conduct experiments with different fixed factors and independent variables. In the context of Math IR, for example, a test collection designed solely for evaluating formula retrieval (i.e.,

with queries taking exclusively the form of formulae) cannot be reused to evaluate joint retrieval of text and formulae. In contrast, a test collection with queries taking the form of natural text with embedded formulae can be used for both evaluations. As we will see in Section 2.4, the completeness of relevance judgements also influences the reusability of a test collection.

### 2.3.2 The Probability Ranking Principle

The *probability ranking principle* (PRP) is the underlying theory that underpins ranking-based probabilistic models (Maron and Kuhns, 1960; Cooper, 1972; Robertson, 1977). The PRP can be stated as follows (paraphrased from Cooper (1972), as cited by Robertson (1977)):

If an IR system responds to a user's request with a ranking of the documents in the collection in decreasing order of probability of relevance, in a way that ensures that this probability has been calculated as accurately as possible using all available information, then the overall effectiveness of the system to the user is the best obtainable given the available data.

Robertson (1977), using material in Cooper's (1972) unpublished paper, has demonstrated the validity of the PRP using two justifications – one through probability theory and another through decision theory. He concludes that in order for the PRP to be applicable in a feasible manner in practice, two fundamental assumptions have to hold in the experimental setup. Both of these assumptions will become important for the design of my test collection.

**Assumption A:** Relevance is binary, i.e., a document in the collection can either be relevant or non-relevant. This assumption enables descriptions of retrieval using theoretical frameworks such as probability and decision theory, and makes their application feasible in practice.

**Assumption B:** Relevance is independent, i.e., the relevance of one document to the user request must not depend on the other documents in the collection. One example given by Robertson (1977) of documents becoming dependently relevant to a query involves two documents each tackling complementary aspects of the problem described in the query. In the context of Math IR, relevance dependence can occur when the resolution of a particular mathematical query requires results from two documents. To assume the PRP therefore means ignoring such cases. The PRP makes large-scale comparative IR evaluation feasible: system document rankings are computed on independent document scores, rather than order-permutations of all documents in the collection.

### 2.3.3 Evaluation Measures

For the purpose of evaluation, retrieval can be considered to be a binary classification task since, in its simplest form, a retrieval system has to decide whether a document is relevant to the query or not. This allows IR researchers to draw a standard confusion matrix, like the one shown in Table 2.1.

|              |              | Predicted Class      |                      |
|--------------|--------------|----------------------|----------------------|
|              |              | Relevant             | Not Relevant         |
| Actual Class | Relevant     | True Positives (TP)  | False Negatives (FN) |
|              | Not Relevant | False Positives (FP) | True Negatives (TN)  |

Table 2.1: Confusion matrix for IR evaluation as binary classification.

Evaluation of retrieval models is usually expressed in terms of *Precision* and *Recall*. Given a query, precision is defined to be the number of *relevant documents retrieved* over the *total number of retrieved documents* in the collection:

$$P = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of retrieved documents}} = \frac{TP}{TP + FP}$$

Similarly, *recall* is the number of *relevant documents retrieved* over the *total number of relevant documents* in the collection:

$$R = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of relevant documents}} = \frac{TP}{TP + FN}$$

Precision and Recall are inversely correlated and are often reported together. The  $F_1$  score can be used to summarise precision and recall in one metric. The  $F_1$  score is the harmonic mean of precision and recall:

$$F_1 = 2 \times \frac{1}{\frac{1}{R} + \frac{1}{P}} = 2 \times \frac{P \times R}{P + R}$$

Precision, recall and  $F_1$  score are applicable to general binary classification. In fact, I will be using these measures to evaluate non-IR tasks in this thesis, namely the gold standard annotation performed in Section 5.1.3, and the performance of machine learning classifiers from Section 5.3.2.

In the context of ranked retrieval, however, precision, recall and  $F_1$  by themselves are inadequate. In ranked retrieval, only the top  $k$  documents can be returned and, ideally, the relevant documents will appear early on in this set. The main problem with the direct use of precision and recall in ranked results comes from the fact that these measures are designed to work with unordered sets rather than ordered lists. In order to overcome this problem, *precision/recall at rank  $k$*  and *precision at recall* can be used to evaluate models that produce ranked results.

**Precision/Recall at Rank:** The idea is that precision and recall are evaluated at consecutive subsets of the ranked list, in increasing order of rank. This gives a formal indication of where in the subset up to rank  $k$  the relevant documents are found (precision at rank) and how many relevant documents are found (recall at rank).

**Precision at Recall:** This measure is used to formalise the relationship between precision and recall in the ranking (precision as a function of recall). If a perfect ranking algorithm existed,

then all relevant documents would be ranked ahead of all irrelevant. As a result, recall would be 1 for all values of precision and a plot of this relationship would produce a graph with a vertical line at recall=1.

Precision at recall is derived from recall/precision at rank values. In most cases, a fixed number of specific recall (thresholded) values are chosen and the precision at these values is sampled from the precision at rank values. Typically, for a given recall threshold, say 0.1, the first occurrence of a recall at rank  $k$  that is between that threshold and its successor (say 0.2) is used to sample the precision at the same rank  $k$ . Other approaches involve taking the set of all recall at rank  $k$  values within the threshold interval and finding a  $k$  in this subset that maximises the precision (interpolation using the max).

**Mean Average Precision (MAP):** MAP is the most commonly used performance measure in IR and is the primary measure used to evaluate IR models in this thesis. For a set of queries  $Q$ , with each  $q_j \in Q$  having  $|\{r_1, \dots, r_{m_j}\}| = m_j$  known relevant documents, the MAP score is given by:

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(R_{j,k}) \quad (2.2)$$

In Formula 2.2, the term “Precision( $R_{j,k}$ )” is the precision after encountering relevant document  $r_k$  in the ranked list of results for query  $q_j$ . MAP samples precision at each relevant document and produces one *average precision* (AP) value for each query. An AP value of 0 is assigned to any relevant document not retrieved (i.e., not in the results set). The set of average values obtained is summarised into a single value by computing the arithmetic mean of average precision across all queries.

In order to illustrate the use of MAP in practice, consider four queries with 4, 4, 2 and 1 relevant documents respectively. Suppose that the result sets presented in Table 2.2 (where “R” denotes relevant while “N” denotes non-relevant documents) have been obtained by an IR system for these four queries:

| Rank    | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---------|---|---|---|---|---|---|---|---|---|----|
| Query 1 | R | N | R | N | N | N | N | N | R | R  |
| Query 2 | N | R | N | N | R | R | R | N | N | N  |
| Query 3 | N | N | N | N | R | N | N | N | N | R  |
| Query 4 | N | N | N | N | R | N | N | N | N | N  |

Table 2.2: Example result sets (ranked) for a test collection with four queries (adapted from exercise 8.8 in Manning et al. (2008)).

Queries 1 and 2 both return 4 relevant documents in the top 10 ranks. However, in terms of

MAPs associated with each query<sup>7</sup>, Query 1's MAP is higher than Query 2's MAP:

$$\begin{aligned}\text{query 1, } AP_1 &= \frac{1}{4} \times \left( \frac{1}{1} + \frac{2}{3} + \frac{3}{9} + \frac{4}{10} \right) = 0.6 \\ \text{query 2, } AP_2 &= \frac{1}{4} \times \left( \frac{1}{2} + \frac{2}{5} + \frac{3}{6} + \frac{4}{7} \right) = 0.493 \\ \text{query 3, } AP_3 &= \frac{1}{2} \times \left( \frac{1}{5} + \frac{2}{10} \right) = 0.2 \\ \text{query 4, } AP_4 &= \frac{1}{5} = 0.2\end{aligned}$$

This illustrates that MAP is sensitive to the entire ranking and rewards systems that retrieve relevant documents early on in the ranked results.

Note also that Queries 3 and 4 both yield the same MAP despite the fact that Query 3 has more relevant documents than Query 4, and returns the second relevant document as late as rank 10. This example illustrates that MAP is not as easy to interpret as simpler measures such as precision at rank, partially because it is dependent on the number of relevant documents per query, which typically varies considerably.

Information Retrieval experimenters are responsible for selecting an evaluation measure that increases confidence in the reliability of their comparative evaluation. An appropriate evaluation measure is one that is best aligned to the objectives and design of the evaluation to be performed. For example, comparative evaluation for a web retrieval task should favour precision over recall, because most users never consider hits further down than the first page of results.

Often, selecting an appropriate evaluation measure involves careful consideration of the theoretical properties and experimental behaviour of evaluation measures. Given two systems  $X, Y$  and two test collections  $T_1$  and  $T_2$ , the following properties have been identified by the IR community as criteria for selecting a measure  $M$  for a particular evaluation scenario:

**Stability:** According to Buckley and Voorhees (2000), measure  $M$  is stable if the evaluation using  $M$  is not significantly changed by perturbations in the expression of information needs between two document collections  $T_1$  and  $T_2$ . In the experiment, they replaced queries with similar formulations of the same queries and measured whether the evaluation metrics changed too much, leaving both document collections and relevance judgements the same. For instance, in ranked retrieval, a stable measure  $M$  would produce similar rankings for  $X$  on  $T_1$  and  $T_2$ <sup>8</sup>.

<sup>7</sup>If we take queries 1 and 2 to be separate test collections with one query each, then we have that  $MAP_1 = AP_1$  and  $MAP_2 = AP_2$ .

<sup>8</sup>In this sense, stability is a special case of the related property "reliability" used by Zobel (1998) as the property of the non-change of the evaluation results if  $T_1$  and  $T_2$  represent a "similar" retrieval scenario.

**Sensitivity:** The ability of a performance measure to detect small differences between retrieval methods<sup>9</sup>. Buckley and Voorhees (2000) explain the concept as follows. Suppose there are  $n$  runs of methods  $X$  and  $Y$  on test collection  $T_1$ . Suppose also that method  $X$  is better (or worse) than  $Y$ . A measure  $M$  is said to have higher discriminating ability than measure  $L$  if the proportion of  $n$  runs in which measure  $M$  determines that method  $X$  is better (or worse) than  $Y$  is higher than those of measure  $L$ . Thus, in the context of a comparison of two systems, it is the probability of determining the best system, given that a difference between the two systems exists. Sensitivity is particularly important when comparing close variants of the same retrieval model, e.g., during development of the model, as these tend to operate very similarly. In ranked retrieval a sensitive measure can detect improvement due to a large number of independent small improvements without the observance of a large, overall improvement. Sensitivity is known to depend on the number of topics (queries) and the number of relevant documents: measures with lower sensitivity require a larger number of topics before a conclusion as to which retrieval method is better can be made.

I will use MAP to measure the performance of retrieval systems in my experiments (Chapter 7). MAP has been shown to be a stable and sensitive measure for comparative evaluation in IR scenarios where there are at least 50 topics and relevance judgements are complete (Buckley and Voorhees, 2000; Buckley and Voorhees, 2004; Voorhees and Harman, 2005).

## 2.4 Methodologies for Building Test Collections

In Section 2.3 I introduced the test collection paradigm for conducting IR experiments in a laboratory environment. A test collection consists of three components: a document set, a query set and a set of relevance judgements. The process of constructing a test collection involves sourcing each of these components and compiling them into a unified resource. In this section, I will present established methodologies for procuring each component, and discuss their advantages and disadvantages.

### Queries

In a test collection, the query set models the innumerable searches a diverse population of users carries out in the real world. There are two approaches to procuring queries: *manufacturing* and *observing*.

Manufacturing queries involves artificially generating queries for a particular document collection. Queries are created by individuals simulating information needs and writing queries with a specific subset of the document collection in mind. This approach ensures that documents and queries are not disassociated and that at least a few relevant documents exists but is time consuming and might not be representative of genuine information needs (violation of Property 1, realismness, from section 2.3.1). There is also the risk of introducing bias in the test collection

---

<sup>9</sup>Some authors refer to this property as “discriminating ability”.

if the queries are created by the experimenters themselves because queries are created by authors who have specific relevant documents in mind.

Alternatively, queries can be produced by observing the users of real IR systems. Expressions of user needs submitted to an existing IR system can be collected and adapted into queries for the test collection. Queries produced by observation are likely to represent genuine, real-life information needs that have actually been experienced by a searcher.

Experimenters can observe the information needs of users by either interviewing the users about their most recent searches (Teevan et al., 2004) or by examining query logs (Jansen et al., 1998). These approaches have disjoint advantages. For example, interviewing users about their information needs allows experimenters to produce queries that accurately represent the original information need through user-interactive refinements. In contrast, query logs of operational systems are samples of real-world queries that should be large enough to be representative of the diversity of queries and users of these systems.

Adapting information needs from observed IR systems for test collection queries requires considerable effort by the experimenters. On one hand, interviewing users of IR systems is not only extremely time-consuming, but can also be disruptive in an operational setting. Therefore, experimenters have limited access to users and must carefully plan user-interactive sessions a priori. On the other hand, experimenters examining query logs do not have access to the observed IR system's users. As a result, the experimenters might have to infer the information needs from submitted queries with limited search context. Consequently, the experimenters must be prepared to defend the test collection against claims that its queries are subjective interpolations of the information needs in the logs.

### **Relevance Judgements**

An important consideration in the procurement of relevance judgements is who is the judge of relevance. Ideally, relevance judgements are made by the author of the query at the time of creation (Clough and Sanderson, 2013; Voorhees, 1998). This is a realistic model of user searches: it simulates the real-life operational setting whereby the query author assesses the quality of the documents returned by the system in real-time.

Alternatively, judgement of relevance can be made by someone other than the author of the query. This approach is not operationally realistic (e.g., does not model a single user's search), but can be a practical compromise when building large test collections. For example, assessment of relevance can be distributed across many assessors who have had nothing to do with the creation of the queries. In this case, it has been found that expert assessors with familiarity in the domain of the queries produce higher quality relevance judgements than non-experts (Bailey et al., 2008). There are two considerations when delegating relevance judgements to assessors other than the query author. First, relevance judgements have been shown to differ between assessors (Voorhees, 1998). However, Voorhees has also shown that differences in relevance judgements

between assessors on the same collection does not significantly affect the relative effectiveness of systems. Second, relevance judgements vary across time – the same assessor might make different relevance judgements at different points in time.

In order to make the test collection reusable (property 3 of test collections, section 2.3.1) relevance judgements must be *complete* – every document in the collection should be judged with respect to every query. This is an observation that was first made by Aitchison and Cleverdon (1963). Complete relevance judgements are necessary for accurate computation of recall and produce stable evaluation metrics (e.g., MAP, as we have seen in section 2.3.3).

Modern document collections are large and may contain several hundred thousand documents. This makes obtaining complete relevance judgements by manual assessment infeasible. Spärck-Jones and van Rijsbergen (1975) proposed a technique for large-scale procurement of relevance judgements known as *pooling*. In the pooling method, the systems participating in the evaluation are used to obtain the top  $n$  results for each query, where  $n$  is referred to as the *pool size*. The results for a given query across all systems are merged with duplicate documents removed. This pooled set is then forwarded to human assessors for manual assessment (Spärck-Jones and van Rijsbergen, 1975; Voorhees and Harman, 2005).

Systems participating in pooling represent distinct retrieval strategies. The pooling method makes use of this fact to obtain a diverse initial set of opinions of relevance in a feasible manner. The workload of human assessors is best reduced by the application of pooling when the number of relevant documents per query is maximised under the smallest possible pool size.

Pooling was adopted by TREC (section 2.4.2) for the construction of the TREC ad-hoc test collection (Harman, 1994; Harman, 1995; Voorhees and Harman, 2005). An important question is to which degree pooled relevance judgements can be considered “complete” and whether they really produce stable results. As we will see in section 2.4.2, pooling in TREC accounted for the majority of relevant documents in the test collection, was not biased towards the systems participating in pooling and produced reliable results overall (Zobel, 1998).

## **Document Collection**

In the test collection paradigm, the document collection simulates the “search space” of the retrieval environment. In real life scenarios, one can encounter dynamic search spaces (e.g., the Web), spaces varying in size, document length, document type and degree of document homogeneity. Although document collections in real-life can grow, laboratory experimentation requires that the document collection be a controlled variable. The collection must be static; otherwise complete relevance judgements cannot be obtained.

The suitability of a document collection depends on the retrieval scenario being simulated (i.e., the task) and must be adequately related to the queries in the test collection. Additional requirements may be imposed by the nature of the experiment or evaluation. For example, investigating the relative effects of document length normalisation will require a test collection

with documents varying in length. In contrast, an experiment designed to evaluate system performance on very specific queries with few relevant documents will require a document collection with high numbers of similar documents. An example of such an experiment is the retrieval of research-level mathematical information needs addressed here: queries are very specific, have few relevant documents and the documents in the collection, although topically diverse, are similar with respect to expertise of the authors and writing style.

Finally, the size of the document collection is also an important consideration. If the real-world scenario being simulated involves searches over large collections of documents, then the test collection should reflect this. Voorhees (2002) notes that larger document collections should be preferred for two reasons:

1. As the number of relevant document increases the evaluation metrics become more stable.
2. Larger document collections are more diverse. This makes the task of retrieval realistically challenging since it accounts for the many ways a concept can be expressed in natural language.

In the next section, I will discuss some established methodologies for constructing test collections.

#### **2.4.1 The Cranfield Methodology**

The test collection paradigm can be traced back to the Cranfield experiments conducted by Cyril Cleverdon and his group at the Cranfield Institute between 1958 and 1966.

##### **Cranfield I**

The Cranfield I experiment (Cleverdon, 1960; Cleverdon, 1962) was designed to be a comparative evaluation of four manual library indexing systems. The experiment encompassed 1,200 search questions and a document collection of 18,000 scientific documents about aeronautical engineering.

Each question for Cranfield I was derived from one document in the collection referred to as the *source document*. Authors of source documents were asked to write a question that is adequately answered by their work (Cleverdon, 1960; Cleverdon, 1962). The source document for each question was also considered to be its unique relevant document. Manual indexing was carried out using a Latin Square Design in order to control for variability in experience between human indexers. The process of retrieving documents for each question involved manual utilisation of each indexing system. The search was considered to be successful once the source document was retrieved.

## **Cranfield I $\frac{1}{2}$**

In parallel with Cranfield I, the Cranfield institute was conducting another study with Western Reserve University (WRU) intended to test a different indexing method (Aitchison and Cleverdon, 1963). This study, later known as Cranfield I $\frac{1}{2}$  (Spärck-Jones, 1981), was carried out using 1,000 documents of the Metallurgical Index at WRU and 104 questions (Aitchison and Cleverdon, 1963; Cleverdon, 1991). According to Cleverdon's own account (Cleverdon, 1991), it was through Cranfield I $\frac{1}{2}$  that he and his team began to realise that recall was of limited use unless precision was also measured. This motivated the team to look for relevant documents in the collection other than the source document. Ultimately, the relevance judgements were extended to include documents that cite the source documents.

## **Criticism of Cranfield I and I $\frac{1}{2}$**

The Cranfield I and I $\frac{1}{2}$  experiments employed what is today referred to as the *source document principle*, whereby a source document is used to (a) derive a query from its topic and (b) the source document is not removed from the test collection but retained as a relevant document during evaluation.

The application of the source document principle has garnered considerable criticism. Swanson (1965) argued that deriving questions from source documents introduces a cognitive bias: question authors are likely to use words and phrases coming from the source document in their formulation of the question. The strong coupling between questions and source documents, Swanson argues, means that detected differences between indexing schemes are unreliable. Instead, Swanson proposed removing the source documents from the test collection and running the tests on "fresh" documents, which would remove undue influences originating from the wording or other features of the question. Sharp (1964) went even further by recommending that the entire source document principle be dropped completely.

The decision of Cranfield I to include only one relevant document per question in the test collection also attracted criticism. Sharp (1964) characterised the decision as an "oversimplification" of retrieval. Swanson (1965) argued that the criterion for judging a search successful, i.e., that the unique relevant document is retrieved, meant that recall was effectively maximised by design (something that Cleverdon had already addressed in Cranfield I $\frac{1}{2}$ ).

## **Cranfield II**

The Cranfield II experiment was a follow-up to the Cranfield I and I $\frac{1}{2}$  experiments (Spärck-Jones, 1981), with the goal of measuring the performance of 29 indexing systems using both precision and recall. The experiment was designed to measure the performance of the systems independently, with operational variables controlled (Cleverdon, 1967; Cleverdon, 1991). In terms of relevance judgements, they were intended to be complete and be procured independently of the questions (Cleverdon et al., 1966; Cleverdon, 1967).

The Cranfield II test collection contained 221 questions and 1,400 documents. It was constructed by asking paper authors to formulate questions on the research problem underpinning their work, which took the form of natural language sentences. Once the authors formulated their questions they were asked to assess the relevance of up to 10 cited references in their to the formulated questions. The questions were submitted back to the experimenters who screened them for grammatical correctness and number of known relevant documents, with 361 questions selected to advance to the relevance assessment stage. Five graduate students with expertise in aeronautics assessed the relevance of 1,400 documents against all 361 questions using a five-point scale. The relevance decisions made by the graduate students for each question were presented to the question's original author for final approval, with a total of 221 questions ultimately approved.

The source document principle was re-used in Cranfield II with two adaptations. First, the source documents were removed from the collection. Second, documents in the reference list of the source documents, including those with relevance assessments by the source document authors, were automatically added in the collection, in response to the above-described criticisms by Swanson (1965) and Sharp (1964).

The influence of question authors on relevance was limited to final approval of the judgements made by the student assessors (unlike in Cranfield I, where the authors themselves made the relevance judgements).

### **Criticism of Cranfield II**

Despite the fact that source documents were excluded from the Cranfield II test collection, the decision to reapply the source document principle received criticism. Vickery (1967) repeated the concerns expressed by Swanson (1965): question authors, having also authored the source document, would subconsciously select words present in the source document when formulating their questions. Vickery also criticised the fact that questions for Cranfield II were formulated after the question authors had read the cited papers, arguing that this might also influence the wording of questions.

### **Significance of the Cranfield Experiments**

The Cranfield experiments introduced test collections as a mechanism for IR evaluation. Five major influences to modern IR evaluation can be traced back to the experiments and have become synonymous to the *Cranfield paradigm* (Voorhees, 2002; Harman, 2011; Clough and Sanderson, 2013). First, Cranfield II introduced test collections as standardised, static and re-usable vehicles for comparative evaluation of IR systems. Second, approximating the operational setting being simulated using real user information needs was a design goal for Cranfield I and II. This approach has been adopted for the construction of modern IR test collections, such as TREC (section 2.4.2). Third, the Cranfield II methodology established the practice of building the test collection prior to conducting the actual retrieval experiment. This ensures re-usability of the

test collection by excluding human bias towards a particular experiment. Fourth, the Cranfield experiments uncovered the importance of using standardised performance measures, such as precision and recall, that relate retrieval performance to how users perceive the quality of results (Voorhees, 2002; Harman, 2011). Fifth, the question of how big a test collection needs to be for results to be statistically significant was first posed and partially addressed by the Cranfield experiments.

The Cranfield experiments made three simplifying assumptions that have become the cornerstone of IR evaluation using test collections. The first assumption is that an infinitely large and diverse set of users and information needs can be modelled using a static, fixed set of queries and relevance judgements. The second assumption is that relevance can be approximated by topical similarity (Voorhees, 2002). This makes documents independently relevant to a query in a test collection: the topic of a single document does not change over time or with respect to other documents in the collection. We have already encountered this assumption as assumption B of the probability ranking principle (p. 36), which means that documents can be scored independently during retrieval. As a result, retrieval and evaluation becomes computationally feasible. The final assumption made by the Cranfield paradigm is that relevance judgements for each query are complete.

#### **2.4.2 TREC**

The Text REtrieval Conference (TREC) is a series of IR workshops, funded by NIST<sup>10</sup>, that introduced the first large-scale test collections for evaluation with common tasks and standard evaluation methodologies (Harman, 1993; Voorhees and Harman, 2005).

##### **TREC Ad-Hoc Retrieval Task**

TREC popularised the Cranfield approach by adapting the test collection paradigm for large-scale IR evaluation in its ad-hoc retrieval task introduced in 1992 (Harman, 1993). As noted in Section 2.3, the TREC ad-hoc task is (a) designed to simulate retrieval of atomic information needs over a large document collection and (b) intended to evaluate a system's ability to retrieve accurate and complete lists of documents in response to these information needs (Voorhees and Harman, 2005).

The first TREC ad-hoc test collection contained 742,611 documents and 50 so-called *topics* (Harman, 1993). Topics were created by retired information analysts with familiarity in using retrieval systems so that the topics would mimic information needs of real IR system users (Harman, 1993; Voorhees and Harman, 2005). TREC topics are abstractions of queries taking the form of multi-part descriptions of information needs relating to a particular subject. Each topic had a standardised structure, a set of named fields, and a unique numeric identifier. Each field was reserved for specifying a particular aspect of the information need. For example, the

---

<sup>10</sup>National Institute of Standards and Technology, United States Department of Commerce.

```

<top>
<num> Number: 396
<title> sick building syndrome
<desc> Description:
Identify documents that discuss sick building syndrome or building-
related illnesses.
<narr> Narrative:
A relevant document would contain any data that refers to the sick
building or building-related illnesses, including illnesses caused by
asbestos, air conditioning, pollution controls. Work-related illnesses
caused by the building, such as carpal tunnel syndrome, are not
relevant.
</top>

```

Figure 2.7: Sample TREC topic

“narrative” field was used to give a detailed description of what constitutes a relevant document for the topic.

The topic design allowed for information needs to be specified in more detail than traditional queries. This level of detail made the information needs behind the topics more understandable to the relevance assessors, enabled more consistent relevance judgements to be made and allowed for further research into query construction methods (e.g., by increasing the information in the topic) (Harman, 1993).

The document collection for the TREC ad-hoc tasks consisted of newspaper and newswire articles (e.g., articles from the Wall Street Journal), patents and government documents (e.g., Department of Energy abstracts and Federal Register documents). This selection of documents models the operational setting the TREC ad-hoc task intended to simulate: a document collection exhibiting diversity in (a) document length (e.g., news articles are shorter than government reports), (b) subject, (c) writing style and (d) vocabulary (Harman, 1993; Voorhees and Harman, 2005).

The ad-hoc task was the main TREC task for eight years. Each year, the test collection for the task used 50 new topics. The size of the document collections was gradually increased to contain between 500,000 and 1,000,000 documents (Voorhees, 2002). The method by which the ad-hoc test collection was constructed was re-applied between conferences and is as follows.

1. TREC topics were created by retired information analysts, who were asked to reverse-engineer topics with specific documents from the collection in mind.
2. The analysts devised a set of initial topics spontaneously or by getting ideas while (a) browsing subject areas of interest in the collection or (b) doing test searches (Voorhees and Harman, 2005). Gradually, instructions for topic creation were refined to ensure that topics were as realistic as possible. For example, in later TREC workshops topic creators were instructed to (a) submit their own, genuine information needs as topics and (b) produce relevance judgements for them.
3. The initial topics were used to perform trial searches in order to select topics which had 25 to 50 relevant documents. This criterion was put in place so both broad and narrow searches

could be modelled using the test collection.

4. TREC used the pooling method (Spärck-Jones and van Rijsbergen, 1975) for obtaining relevance judgements, with pool size set to 100 documents. The pool set for each topic was forwarded to (possibly different) human assessors, also hired analysts, tasked with making relevance judgements. In order to prevent biasing the relevance judgements, the assessors did not have knowledge of how the participating systems ranked or scored the documents in each pool set (Voorhees and Harman, 2005).
5. Assessors were asked to judge a document as relevant if the information it contained could be used to write a report on the topic. The relevance judgements for a particular topic were made by a single assessor for consistency of judgements (Harman, 1993). Documents not judged to be relevant during pooling were assumed to be non-relevant, effectively making relevance judgements for TREC ad-hoc binary.

### **Significance of the TREC ad-hoc task test collection methodology**

The TREC ad-hoc task demonstrated how the Cranfield paradigm can be used to build large-scale test collections that are realistic, reusable and diverse (Section 2.3.1). TREC adopted the Cranfield paradigm by (a) building the test collection prior to the evaluation phase of the task, (b) creating topics that model real user information needs and (c) aiming to approximate complete relevance judgements, by implementing the pooling method.

The TREC ad-hoc test collections allowed researchers to test the assumption that pooling produces close to complete relevance judgements. This assumption was tested experimentally at TREC-3 using data from TREC-2 (Harman, 1994; Harman, 1995) and by Zobel (1998).

The TREC-3 study investigated the completeness of pooled relevance judgements by extending the pool from 100 to 200 for 18 ad-hoc topics (Harman, 1995). Relevance judgements were re-done by the original judges for the topics. Harman (1995) reports that on average, 30 new relevant documents were discovered per topic but the distribution was skewed; topics with many relevant documents tended to have many more. When comparing results, it was found that the newly discovered relevant documents were not enough to affect system rankings. The process was replicated with TREC-3 data showing that there was less than one new relevant document per topic on average (Harman, 1995; Harman, 2011). Zobel (1998) estimated that pooling in TREC accounted for about 50-70% of relevant documents. Furthermore, he found that relevance judgements obtained through TREC pooling were not biased towards systems that did not contribute to the pool and concluded that the TREC methodology produces reliable results overall.

#### **2.4.3 Yahoo! Chiebukuro**

The National Institute of Informatics (NII) in Japan has sponsored the NTCIR (“NII Test Collections for IR Systems”) series of IR workshops. So far, there have been 14 NTCIR

conferences<sup>11</sup> which have produced test collections similar in size and format to TREC for tasks such as ad-hoc retrieval (Kando et al., 1999), patent search (Iwayama et al., 2003), question answering (Kato and Masui, 2003) and text summarisation (Fukushima and Okumura, 2001).

The community QA task (Ishikawa et al., 2010) was introduced at NTCIR-8 to simulate the retrieval of community-type QnA (Question and Answer) content. The main objective of the task is “best answer estimation”: given a set of questions, a retrieval system must identify the best answers to those questions. Therefore, queries for the task are questions posted on the site with documents taking the form of candidate answers to these questions.

Ishikawa et al. (2010) used “Yahoo! Chiebukuro” data to observe the community and to construct the test collection for the task. Yahoo! Chiebukuro is a QnA website similar to StackOverflow<sup>12</sup> (a QnA website for common coding tasks) and MathOverflow (the QnA website for research-level mathematics which I will introduce in section 3.2.2).

Members of the Yahoo! Chiebukuro community post questions on the site in one of 285 categories; on the website, the categories are ordered by popularity (number of views or number of questions). Each question, along with related comments and candidate answers, is grouped into one discussion unit called a “thread”. A question can receive more than one candidate answer and the questioner (i.e., the member posting the question) selects as the best answer the answer that is the most convincing and/or satisfying (Ishikawa et al., 2010). However, if the questioner has not identified an answer that meets this criteria, the best answer is the one with the most up-votes from the website’s users.

The NTCIR-8 Community QA test collection was sampled from Yahoo! Chiebukuro data containing 3,116,008 questions plus 13,477,785 answers (3,116,008 acknowledged best answers plus 10,361,777 non-selected candidate answers). Starting from this data set, the NTCIR-8 Community QA test collection was constructed as follows (Ishikawa et al., 2010):

1. 1500 questions were sampled at random from 15 out of the 17 top categories (excluding the topics “Chat” and “Others”). The number of questions to be sampled from each category was determined using the formula

$$\text{distribution}_{\text{cat}} = \text{int}\left(1500 \times \frac{\# \text{ of questions in category “cat”}}{\# \text{ of questions in all categories}} + 0.49\right)$$

with the actual questions being chosen at random.

2. For each of the 1,500 sampled questions the answer selected by the author of each question was recorded and automatically considered to be relevant.
3. To avoid biasing the assessors, Ishikawa et al. shuffled the list of 1,500 answers randomly prior to manual relevance assessment, because answers in the Yahoo! Chiebukuro data set

---

<sup>11</sup><http://research.nii.ac.jp/ntcir/publication1-en.html>

<sup>12</sup><https://stackoverflow.com/>

are sorted based on user preference.

4. Four students with comparable experience in using the Yahoo! Chiebukuro website were employed to manually identify additional good answers for the selected questions. The students assessed answers to the 1,500 questions, using a binary scale, and 7,443 answers using a scale from A to C. The scale is interpreted as follows: A – answer is “satisfactory” to the question, B – answer is only partially relevant and C – answer is unrelated to the question. Each assessor examined every question in the test collection. Inter-assessor agreement was measured using the kappa statistic and was found to vary across categories from “slight” ( $\kappa$  between 0.0 and 0.2) to “moderate” ( $\kappa$  between 0.41 and 0.6) (Ishikawa et al., 2010).

As a result of this process, the NTCIR-8 Community QA test collection contains 1,500 questions (queries), each with one best answer (known relevant document). Another 5,943 candidate answers (known less relevant documents) accompany the selected questions and have been included in the test collection as part of its document collection (Ishikawa et al., 2010).

### **Significance of the NTCIR Community QA**

The Yahoo! Chiebukuro website can be considered to be an IR system since its primary function is to answer information needs by eliciting answers from the online community. This makes the Yahoo! Chiebukuro data a snapshot of a real IR system. As a result, questions procured for the NTCIR Community QA task represent genuine and diverse information needs. Furthermore, the primary relevance judgements for the test collection are binary, complete and have been procured by observing the reaction of the questioners themselves (adherence to principles 1, 3 and 4 of section 2.4.3).

The NTCIR community QA task run on the basis of this data involved 4 system and 3 baselines, one of which performed surprisingly well (Sakai et al., 2010). I adopt a similar process (described in section 3.2.4) to observe the MathOverflow online community (section 3.2.2) for the construction of my test collection of research-level mathematical information needs.

Let me recapitulate the most important principles about designing a test collection that the discussion up to now has lead to:

**Principle 1: (Searcher is Ultimate Judge of Relevance, page 41)** *Ideally, the query creator should be the judge of relevance. Final judgement should be made as close to the time the query is created as possible.* This is because the definition of relevance is based on the satisfaction of a searcher’s information need.

**Principle 2 (Probability Ranking Principle; assumption B):** *The relevance of any one particular document to the query is independent of all other documents.* Since the Cranfield experiments, this has been achieved by defining relevance as topical

similarity: whether a particular document is relevant to a query is dependent on the topic of that document alone and is independent on which other documents in the collection are relevant to the same topic.

In the research mathematics retrieval task I am proposing, principle 2 is sometimes violated. Answering research-level mathematical queries often involves some degree of reasoning and inference: the answer might be a combination of many mathematical results (e.g., theorems) distributed in multiple documents. In this case, multiple documents might present complementary aspects of a mathematical problem making them relevant only when seen together. In terms of evaluation using test collections, this is not desirable as it invalidates the PRP (Robertson, 1977). An effective test collection of research-level mathematics must therefore account for principle 2 explicitly as part of the construction process. I will come back to this issue in section 3.2.

**Principle 3 (Probability Ranking Principle assumption A + Staticness (page 42)):**

*Relevance is binary and constant – every document in the document collection is permanently either relevant or non-relevant to every query.*

**Principle 4: (Outcome of Cranfield I,  $I\frac{1}{2}$ )** *Relevance judgements should be complete – every document in the collection should be judged with respect to every query.*

Complete relevance judgements are necessary for accurate computation of recall (section 2.3.3): the computation depends on accurately counting the number of false negatives (not retrieved but relevant). More importantly, as we have seen in section 2.3.3, evaluation metrics such as MAP are stable when computed over complete relevance judgements. Thus, principle 4 serves to make test collections reusable (property 2 of test collections, section 2.3.1).

## 2.5 The Language and Discourse of Mathematical Content

In his study of the language of mathematics, Ganesalingam (2008) notes that written mathematics, unlike general text, follows strong domain-specific conventions that govern how content is presented. The language of mathematics is formal, i.e., more constrained than general natural language, consisting primarily of assertions about mathematical objects and facts, which can be either true or false.

Mathematicians treat mathematical objects as “timeless constructs” that “exist independently of the physical world” (i.e., are platonic ideals) eliminating the need to describe events. Therefore, assertions are written as sentences in the present simple tense. Typically, such sentences are constructed using passive form or first person plural using a limited number of verbs (a potentially closed set of verbs). Morphological variation is likewise limited, except for number and mood, which are the only morphological dimensions to demonstrate the full range of variation seen in general natural language.

With respect to syntax, Ganesalingam argues that textual mathematics exhibit relatively little variation. He characterises textual mathematics as a “formulaic language” and notes that the absence of long-distance dependencies together with the limited range of morphological phenomena imply that textual mathematics can be captured effectively using a context-free grammar (CFG).

In contrast, the vocabulary of textual mathematics is extraordinarily varied, adaptable and productive. The lexicon is comprised of two types of lexemes: mathematical and extra-mathematical. Mathematical lexemes (i.e., technical terms) constitute the bulk of the lexicon.

A small number of extra-mathematical lexemes are words such as “hence”, “denote” and “suppose”, which are used as connectives between arguments. In addition, there is a certain number of extra-mathematical phrases. Interestingly, these phrases are particularly rigid and even their substitution with synonyms is resisted by mathematicians. This, Ganesalingam argues, enables us to consider them as fixed lexemes, or “set-phrases”. Examples of such set-phrases include “without loss of generality”, “it is easy to show that”, “it follows that” and “it remains to prove that”.

Ganesalingam emphasises that the meaning carried by textual mathematics can be described by a small subset of the machinery necessary to model general natural language. Since textual mathematics describe timeless objects, there is no need to consider tense or model the description of events. Due to the absence of presuppositional attitude reports, typically used to talk and make assertions about state of mind, there is no intensionality in textual mathematics. Additionally, the truth-status of mathematical assertions is irrespective of the opinion of the speaker. As such, there is no need for modality in textual mathematics. Similarly, textual mathematics is intended for literal interpretation eliminating any expectation of irony or metaphor. Ganesalingam concludes that standard semantic representations could potentially capture the meaning of textual mathematics, unlike what is the case for general natural language.

In mathematical text, two contexts interact to convey the meaning: the *textual context* and the *symbolic context*. The textual context is the part of written mathematics that resembles natural language text while the symbolic context refers to the symbols and notation that are interspersed in the mathematical discourse. Symbolic material is used to abbreviate statements and assertions to make the material easier to read. For example, in number theory the expression  $d|n$  abbreviates the statement

“ $d$  and  $n$  are integers such that the remainder of the ratio  $\frac{d}{n}$  is also an integer” (i.e.,  $n$  divides  $d$ ).

Ganesalingam notes that due to its abbreviative role, the symbolic context tends to be embedded inside the textual context (i.e., appears inside text like “islands”) with symbolic elements often replacing noun phrases. There is strong interdependence between the textual and symbolic contexts in terms of both syntax and semantics – text without symbol is incomplete while

symbol without text is senseless (i.e., what the symbols abbreviate cannot be deduced) and incomprehensible (e.g., argumentation is missing). For example, consider the following sentences with their symbolic context removed:

Let  $@@@$  be a real function. For  $@@@$ ,  $@@@$  is undefined...  
Suppose  $@@@$ . Then,  $@@@$  and  $@@@$ . Therefore,  $@@@$ .

Removing the symbolic context from these sentences renders them incomprehensible because the reader is missing information regarding what is being manipulated and what properties are being attached to the intended mathematical constructs. On the other hand, without the text, the reader can infer neither the denotation of the symbols nor the overall argumentation that is being made at the sentential context:

~~Let  $x$  be a vector.  
If  $b^{-1}$  is Pisot, then there is an ergodic measure of maximal dimension.~~

Making sense of either the symbolic context or the textual context alone is not sufficient. For one, the symbolic context of scientific documents is inherently ambiguous. Examples of ambiguities range from implicit multiplications, unary operators having irregular precedence as well as overloaded terms and operators (with precedence and associativity properties implicitly represented). Furthermore, the definitions and overall sense of terms used in the symbolic context may be defined in the textual context and vice versa.

Ganesalingam has shown that ambiguities in either context, although hard to resolve by examining each context in isolation, can be addressed successfully when both are examined in unison and when the notion of a *type* is introduced. However, as Ganesalingam demonstrates, the traditional notion of a type based on property membership which translates to some logical predicate is problematic in this context. In particular, the traditional notion of a type breaks down in language instances where compositionality is not applicable. Additionally, equality based on property membership is subject to undecidability (i.e. deciding the type of a piece of mathematics may involve solving the problem itself). In order to overcome these problems, Ganesalingam introduces a type system that is non-extensional (where types act more or less as labels), facilitating the unification of syntactic analysis with type inference. This unified approach enables disambiguation of both contexts simultaneously with type information flowing from one context to the other. In Chapter 4, I will come back to Ganesalingam's idea of "types" and adapt the notion for mathematical information retrieval.

The flow of mathematical content can be interrupted by mathematical rhetoric structures such as mathematical blocks (results, proofs and definition blocks and statements), variables or sets of assumptions. But despite this, Ganesalingam highlights that the discourse of mathematical literature can be modelled linearly (a total order of mathematical knowledge). This will become

important when I consider how to sample sentences from the mathematical discourse in order to build a machine learning data-set for my variable typing task (Chapter 5).

## 2.6 Summary and Approach

It is fair to say that current MIR models are more interested in formula retrieval than in modelling the language of mathematics. A standard approach to MIR is to see it as “special treatment of formulae plus standard IR handling of language”. In particular, joint models of formulae and text have remained largely unexplored despite the fact that some kinds of mathematical information are encoded in parallel in both text and formulae. Current MIR systems typically compute scores for the textual and mathematical contexts independently and then combine them.

The bag-of-words paradigm in particular seems to be ill-suited for textual mathematics (Larson et al., 2013). Mathematical types, which occur plentifully in mathematical writing, should be a useful source of information for retrieval because the dictionary of technical terminology in mathematics is open and productive (Ganesalingam, 2008). But under the bag-of-words paradigm, noun phrases are broken up into their constituents and treated as orthogonal streams of information. Experimental evidence from Larson et al. (2013) supports the hypothesis that this is damaging; the authors stated that performance was “terrible”, despite their extensive efforts to manually expand queries with meaningful, related keywords and went on to conclude that keyword-based retrieval of mathematical text is inadequate for MIR. In chapter 4 I argue for treating mathematical types as atomic units of information, and as first-class citizens in other ways too.

In section 2.5, we saw that mathematical sentences also contain many mathematical set phrases from a small, fixed dictionary, meaning that they are likely to occur often and be present in many documents in a mathematical corpus. In chapter 5, I propose a supervised ML framework that uses these fixed phrases for assigning variables to their types.

Nghiem et al. (2014) state that mathematical expressions are highly abstract and rewritable representations – the same mathematical ideas can be represented using many semantically equivalent but structurally different formulae. For example, two synonymous expressions might exhibit different structure (e.g.,  $y = x^{2+g}$  and  $x^{g+2} = y$ ) while structurally similar expressions might represent different mathematical concepts altogether. This makes structural similarity alone insufficient as a model for semantic similarity of mathematical expressions. In particular, methods able to exploit the interaction between textual and symbolic contexts, thus performing *joint retrieval*, would be attractive. I want to study whether and how information in the textual context can semantically enrich information encapsulated in the mathematical context.

In this thesis I investigate two hypotheses related to the retrieval of research-level mathematics. My first hypothesis is that the textual context of research-level queries and mathematical papers is an underutilised source of information. I hypothesise that technical terminology in mathematical

text, used to label mathematical objects and concepts, has higher discriminating power than individual words when it comes to retrieving research-level mathematical information needs. My second hypothesis is that joint modelling of text and formulae in the mathematical discourse can improve retrieval of research-level mathematics.

The focus of my investigation is retrieval of *research-level* mathematical information needs. This approach allows me to overcome the challenges related to large-scale evaluation of MIR models outlined in Section 2.2. As we have seen in this chapter, evaluation in MIR often involved researchers creating a small test collection specifically for evaluating a new method. Despite the high-quality of the recently introduced NTCIR MathIR test collections (Aizawa et al., 2013), Guidi and Coen (2015) identify some shortcomings: (a) performance is inconsistent between different versions of the test collections and (b) too much emphasis is placed on exact formula matching, rather than document retrieval as a mixture of text and expressions.

I propose that real-life, research-level queries be procured from MathOverflow, an online QnA platform that enables practising mathematicians to post questions arising from their research to a community of graduate and research mathematicians. In Chapter 3 I describe my process for constructing an MIR test collection using questions from MathOverflow as queries, and papers cited in approved answers as relevance judgements. My test collection is large and is composed of real-life queries that are the product of active mathematics research. More importantly, queries in my test collection have been created by mathematicians of roughly the same mathematical background (e.g., graduate-level research) and are thus relatively uniform in their level of abstraction.

In Chapter 4, I introduce my notion of “mathematical types”, which is central to my investigation of both hypotheses. Mathematical types are technical terms used to refer to mathematical objects and concepts in the mathematical discourse. I use types to (a) overcome the limitations of keyword-based search in MIR, (b) enrich formulae with information from the text in the form of denotations for variables and (c) to establish a link between the textual and symbolic contexts for joint retrieval.

In the light of extremely bad performance of keyword-based retrieval methods in MIR, my approach to mathematical text retrieval treats types as atomic units of information. The hypothesis behind this is that the individual words making up type phrases should not be regarded as orthogonal sources of information, as bag-of-words retrieval does. My experiments using query expansion based on mathematical types presented in section 7.1 show that this approach is significantly better than state-of-the-art text retrieval based on the bag-of-words approach.

In Chapter 5, I introduce the variable typing machine learning task designed to assign denotations (types) from the textual modality to variables in the symbolic modality. The link established by variable typing allows me to develop a form of joint retrieval (text and formula retrieval) that I refer to as typed retrieval.

In Chapter 6, I develop typed retrieval models as adaptations of tree matching algorithms and

the Tangent structural heuristic. My adaptations measure formula similarity between expressions by examining the denotations of their constituents, rather than their raw symbols.

My intuitions about type-based textual and typed retrieval are put to the test in Chapter 7. In Section 7.1 I test retrieval methods that are textual, i.e, that give no special status to formulae, but that are allowed to use the type information. One way this can be achieved is to up-weight occurrences of types (phrases) in text; a more sophisticated approach is to expand queries with types that are similar to those present in their text (query expansion).

Models in section 7.2 do the opposite: formula retrieval is handled by task-specific methods while textual information is retrieved by traditional, term-based approaches. In section 7.3 I evaluate my full joint retrieval models: formula retrieval makes use of typed constituents and retrieval of text is type-aware.

Despite the sophistication of these models and the fact that joint retrieval draws information from both modalities, the best performing model is one that relies on textual type information (introduced in Section 7.1). Another high performing model is a version of tree matching that incorporates a scoring component that measures the overlap of types assigned to the constituents of formulae through variable typing (Chapter 5), type disambiguation (Section 6.3.1) and SLT typing (Section 6.3.2).

In chapter 8, I will draw conclusions from my experiments, discuss what I consider to be my own discoveries and identify future work.

# Chapter 3

## A Test Collection of Research-Level Mathematical Information Needs

This chapter will discuss a novel method for creating test collections, something I see as one of the major contributions of this thesis. For my work I need a test collection which is composed of real-life, research-level information needs, expressed in the natural language of mathematics. Section 2.4 will argue that existing test collections for mathematical information retrieval (MIR) are not suitable for my investigation. My proposal for a new methodology for test collection creation (section 3.2) is based on a systematic observation of the MathOverflow (MO) online community. The resulting test collection has been released to the community as the *Cambridge University MathIR Test Collection* (CUMTC) (Stathopoulos and Teufel, 2015).

### 3.1 Motivation for a New MathIR Test Collection

In the early days of MIR, the MIR community has been fragmented in terms of common evaluation resources: each research group evaluated their MIR systems using purpose-built test collections (section 2.2). The NTCIR math IR tracks addressed this fragmentation by producing shared and standardised data sets for comparative evaluation of MIR systems (Aizawa et al., 2013; Aizawa et al., 2014).

The first shared math IR track was introduced by NTCIR in NTCIR-10 (Aizawa et al., 2013) with a test collection consisting of 100,000 documents from mathematics, physics and computer science, containing a total of 35.5 million formulae. The data was obtained from the arXMLiv project (Stamerjohanns and Kohlhase, 2008; Stamerjohanns and Kohlhase, 2009; Stamerjohanns et al., 2010). The documents were formatted using HTML with embedded MathML. Formulae were converted from  $\text{\LaTeX}$  to MathML using the LaTeXXML tool<sup>1</sup>. The NTCIR-10 math IR track had the following retrieval sub-tasks (Aizawa et al., 2013):

---

<sup>1</sup><http://dlmf.nist.gov/LaTeXML>

1. **Formula task:** The input is a set of formulae queries which may include wild-cards as well as concrete variable symbols. Symbolic queries are expressed using presentation and content MathML (section 2.2.1), with adaptations to enable wildcard variables. For example, the symbolic query

$$\frac{?f(?v+?d)-?f(?v)}{?d}$$

taken from Aizawa et al. (2013) includes 3 wild-card variables:  $?d$ ,  $?f$  and  $?v$ . Aizawa et al. (2013) highlight that MIR systems running this query should match

$$g'(cx) = \lim_{h \rightarrow 0} \frac{g(cx + h) - g(cx)}{h}$$

since it contains the query as a sub-expression through the substitution:  $?d \mapsto h$ ,  $?f \mapsto g$  and  $?v \mapsto cx$ .

2. **Full-text search:** Queries take the form of a combination of keywords and formulae. For example<sup>2</sup>:

**NTCIR10-FT-7: Roots of a Polynomial**

**Query:**  $8x^3 + 4x^2 - 4x - 1$

**Words:** root

**Relevance:** The open problem at <http://planetmath.org/?op=getmsg;id=22097> could use a hint. Can we match this to  $x^3 + ax^2 + bx + c = 0$ ?

There were also plans for a third task in NTCIR-10 (Open Information Retrieval task), but it was cancelled due to an operational problem and not continued in NTCIR-11.

The NTCIR-10 math track included 22 formulae queries and 15 full-text queries, procured from mathematicians briefed in the query format, but there is no further information on how the queries are produced (Aizawa et al., 2013).

Six systems participated in the task and were used to obtain an initial set of relevance judgements using the pooling method (pool size set to 100). Further relevance assessments on the pooled sets were produced by mathematicians from Zentralblatt Math and math students from Jacobs University in Germany. Relevance judgements were made based on the formulae in the document: each formula in the topics was judged against each formulae in every document.

Topics were judged by one or two assessors using three grades: R (relevant), PR (partially relevant) and N (not relevant). Since some topics were judged by one assessor while others by two, the final relevance grade was determined by the scheme presented in Table 3.1 (adapted from (Aizawa et al., 2013)).

For NTCIR-11, the main subtask (“ad hoc retrieval with formulae keyword queries”) was continued. 50 formulae/keyword topics were used, and they emulated the format of TREC:

<sup>2</sup>Full-text search topic 7 taken from the topic statement document at <http://ntcir-math.nii.ac.jp/wp-content/blogs.dir/13/files/2014/02/NTCIR10-math-topics.pdf>

| Score | Assessed by one judge | Assessed by two judges | Overall grade      |
|-------|-----------------------|------------------------|--------------------|
| 4     | R                     | R/R                    | Relevant           |
| 3     | -                     | R/PR                   | Relevant           |
| 2     | PR                    | PR/PR,R/N              | Partially relevant |
| 1     | -                     | PR/N                   | Partially relevant |
| 0     | N                     | N/N                    | Not relevant       |

Table 3.1: Relevance grades for the NTCIR-10 Math IR track.

each topic contained three fields (a) topic ID, (b) a query (formula + keywords, no natural language description) and (c) a “narrative” field that described the use-case and defined the relevance criteria. Like in the NTCIR-10 math track, the formula in topics were represented using presentation/content MathML (section 2.2.1) and included wildcards.

The NTCIR-11 document collection was expanded to 105,120 scientific articles (mathematics, physics and computer sciences) from the same source, ArXiv. Documents were segmented into paragraphs, with each paragraph considered to be a “return unit” – the unit of retrieval is a paragraph, rather than the document itself. This resulted in 8,301,578 search units with about 60 million formulae (Aizawa et al., 2014).

Relevance judgements for the main NTCIR-11 track were produced using pooling (eight systems and pool size set to 50). Like in NTCIR-10, two assessors (third-year or graduate mathematicians) made judgements on the relevance of pooled search units. The relevance judgements were binarised using the scheme in Table 3.1.

In NTCIR-11, two Wikipedia tasks were added. The Wikipedia Formula task contained 30 topics with at most 4 keywords. The Wikipedia Open sub-task that focused on retrieving formulae from Wikipedia articles using 100 formula queries selected at random from Wikipedia. Systems participating in the optional task were not formally evaluated with relevance judgements. Instead, a final judgement on performance was made based on a presentation delivered by each participant (Aizawa et al., 2014).

Guidi and Coen (2015) praised the NTCIR math IR tracks for the provision of shared and standardised data sets for comparing MIR systems, but criticised the fact that their test collections over-emphasise formula pattern matching. Guidi and Coen concluded that the state of the NTCIR test collections at the time of their introduction did not encourage “investment into deeper extraction of semantics” from indexed documents.

My goals in this thesis, as outlined in section 1, are two-fold: (a) develop retrieval models for real, research-level mathematical information needs and (b) investigate the effects of jointly modelling text and formulae for mathematics retrieval. In contrast, the NTCIR test collections are designed to simulate the operational setting of general MIR with emphasis on formula search (Aizawa et al., 2013; Aizawa et al., 2014). Following from the differences in operational setting between these two tasks, there are several reasons why I cannot use the NTCIR test collections for my investigations.

1. It is unclear whether the topics in the NTCIR test collections represent genuine research-level needs of real users that arose during the process of conducting mathematics research.
2. The topics in the NTCIR test collections contain information needs that are diverse with respect to the difficulty of their underlying mathematical problems. Some topics are broad, like full-text search topics 7 and 14 below, while others are more specific, like topic 5.

**Topic 7: Roots of a Polynomial**

**Query:**  $8x^3 + 4x^2 - 4x - 1$

**Words:** root

**Relevance:** The open problem at <http://planetmath.org/?op=getmsg;id=22097> could use a hint. Can we match this to  $x^3 + ax^2 + bx + c = 0$ ?

**Topic 14: Convergence**

**Query:**  $\sum p_n a_n$

**Words:** convergence

**Topic 5: Radius of Convergence**

**Query:**  $\sum \frac{n!x^n}{n^n}$

**Words:** radius of convergence

**Relevance:** The open problem at <http://planetmath.org/?op=getmsg;id=23642> could use a hint.

In contrast, the operational setting of my task is characterised by information needs that are very specific, and much more homogenous with respect to their level of difficulty: as they concern research-level mathematics, they are all equally hard to answer.

3. Because many topics in the NTCIR test collections are broad, the collections have many relevant documents per topic. In contrast, as we will see in section 3.2.5, the task of retrieving research-level mathematical information is characterised by topics that have few relevant documents – a consequence of their specificity.
4. The NTCIR test collections contain at most 50 topics that are diverse in terms of difficulty. In contrast, my task is characterised by topics of high specificity with few relevant documents (section 3.2.5). Therefore, a larger number of topics might be necessary to accurately measure the effectiveness of retrieval models for my task.

Second, the NTCIR test collections are designed with assumptions that are incompatible with my investigation for the following reasons:

1. My first hypothesis is that mathematical types – technical terms that refer to mathematical concepts – are more important than normal terms when it comes to retrieving research-level

mathematical information needs. The NTCIR test collections assume that MIR queries are bags of words and formulae, given in the **Query** and **Words** subfields. This approach is incompatible with investigating my first hypothesis because types in the NTCIR topics are broken up into their constituent words, with each word considered to be an independent, orthogonal source of information.

2. My second hypothesis is that joint modelling of text and formulae improves retrieval effectiveness of research-level mathematics. Joint retrieval is preconditioned on the ability to link information from the text to formulae (Schubotz et al., 2016). My idea is to link information from the textual and symbolic contexts through variable typing. Modelling this interaction using machine learning is dependent on the availability of queries expressed using the natural language of mathematics; these tend to have formulae embedded in the query text.

In contrast, topics in the NTCIR test collections are expressed as bags of keywords and formulae. This means that the topics do not have well-formed sentences and are missing important variable typing signals such as copula verbs.

3. The NTCIR test collections assume that the most important component of an MIR retrieval model is formula matching. This motivated the inclusion of formulae with wildcards in their topics so that the pattern-matching abilities of MIR systems can be tested extensively.

I have a different philosophy when it comes to the design of query languages. I assume that users prefer to describe their problem using natural text and concrete formulae instead of using a more expressive but ultimately cumbersome query language with features such as wildcards. My research interest is therefore in enabling MIR systems to match formulae based on syntactic and semantic equivalence in an unsupervised manner, a task in which syntactic peculiarities such as wildcards would be a distraction.

For these reasons I decided to build a new test collection of research-level mathematical information needs.

### **3.2 Cambridge University Mathematics Test Collection (CUMTC)**

I will here present the Cambridge University Mathematics Test Collection (CUMTC), my newly created test collection for research-level mathematical retrieval. It is created based on the idea that topics and relevance judgements can be procured from the on-line collaboration website MathOverflow, a QnA site for practising mathematicians and researchers engaged in mathematics research, through a process of observation.

The construction of my test collection is an apply-once process that results in a test collection that can be reused many times, and it is designed so that it can be replicated by individuals without domain expertise in research mathematics (e.g. at different points in time or in different languages, as long as a comparable resource to MathOverflow exists).

### 3.2.1 Document Collection

As the underlying document set for my test collection, I used the Mathematical Retrieval Corpus (MREC)<sup>3</sup> (Liška et al., 2011). The MREC is a subset of the *ArXMLiv corpus* (Stamerjohanns and Kohlhase, 2008; Stamerjohanns and Kohlhase, 2009; Stamerjohanns et al., 2010) and contains over 439,000 arXiv research pre-prints from mathematical disciplines (e.g., Mathematics, Physics, mathematical Physics, Computer Science and Statistics). While both ArXMLiv and MREC are based on the preprint archive arXiv.org, the MREC only includes those documents from ArXMLiv whose formulae could be converted from L<sup>A</sup>T<sub>E</sub>X to MathML without errors, resulting in documents in XHTML with MathML expression inclusions.

In the context of the task of ad-hoc retrieval of research-level mathematical information needs, the MREC meets the suitability criteria for document collections identified in Section 2.3.1 (page 34), at the beginning of Section 2.4 (page 42) and in Section 2.4.3 (page 50). Specifically:

1. The MREC is large and diverse because it contains hundreds of thousands of documents from many mathematical disciplines. The size and diversity of the MREC contribute to the realism and reusability of my test collection: the underlying document collection can be expected to account for the many ways mathematical concepts can be expressed using natural language.
2. The nature of MIR imposes the additional requirement that evaluation resources for the task must contain many formulae so that their impact on indexing and retrieval models can be studied. The MREC meets this requirement since the documents it contains encode formulae in the form of machine-readable MathML, and there are certainly many of them: 153,796,520 mathematical expressions in MathML format according to my count.
3. Documents in the MREC, by virtue of being preprints, are comparable to the questions and answers in MathOverflow, in terms of topic, level of mathematical abstraction/difficulty and author background. In addition, the arXiv.org repository is frequently cited by users on MathOverflow.
4. MO questions and MREC documents in the MREC were selected independently of material on MathOverflow; the link between MREC documents and MathOverflow threads is established through a naturally occurring process which I as the MIR experimenter simply observe. This ensures that I could not have introduced any bias into the test collection construction.

---

<sup>3</sup>version 2011.4.439

### 3.2.2 MathOverflow

MathOverflow (MO) is an online question and answer site for advanced mathematics with a user base that includes researchers and professional mathematicians (Martin and Pease, 2013; Tausczik and Pennebaker, 2012) – as opposed to [math.stackoverflow.com](http://math.stackoverflow.com), which is for basic mathematics and thus has a very different target group.

MathOverflow is primarily an academic community that views the site as a professional outlet for doing mathematics research, with many published journal articles or pre-prints acknowledging contributions made on MO (Tausczik et al., 2014). MathOverflow users are thus incentivised to make meaningful contributions and develop collaborations on the site because it allows them to form real-world professional relationships and to build professional reputation on MathOverflow (Tausczik and Pennebaker, 2011; Tausczik and Pennebaker, 2012; Martin and Pease, 2013).

A MO user can post a question, usually relating to a small niche field in Mathematics or mathematical Physics, and hope to obtain insightful answers, useful examples and pointers to the literature as well as constructive comments. In response, other users can either post a candidate answer, comment on the question or comment on existing answers. Registered users can vote to close a thread, or vote posts (questions and answers) up or down to express their approval or disapproval, respectively, of the posted material.

A MO thread consists of a question and many answer posts, one of which is designated as the (best) answer. The MO community uses a reputation-based scheme to reward users who produce high-quality answers. Users on the site accumulate reputation through up-votes and by unlocking badges and special achievements when helping other users. Tausczik and Pennebaker (2011) found that the answerer's offline (number of publications in peer reviewed journals) and online (sum of up-votes on all posted answers) reputation are good predictors of MathOverflow answer quality. Furthermore, they found that on average the expertise of answerers (measured as the average of online and offline reputations) exceeded that of questioners.

An informal set of “house rules”, which are enforced by moderators, requires users to express precise, unambiguous research-level questions on the site<sup>4</sup>. MO users acting as moderators ask users to make corrections or clarifications to their posts.

A MO thread's question and designated answer is illustrated in Figure 3.1.

---

<sup>4</sup><http://web.archive.org/web/20130501231334/http://mathoverflow.net/faq>

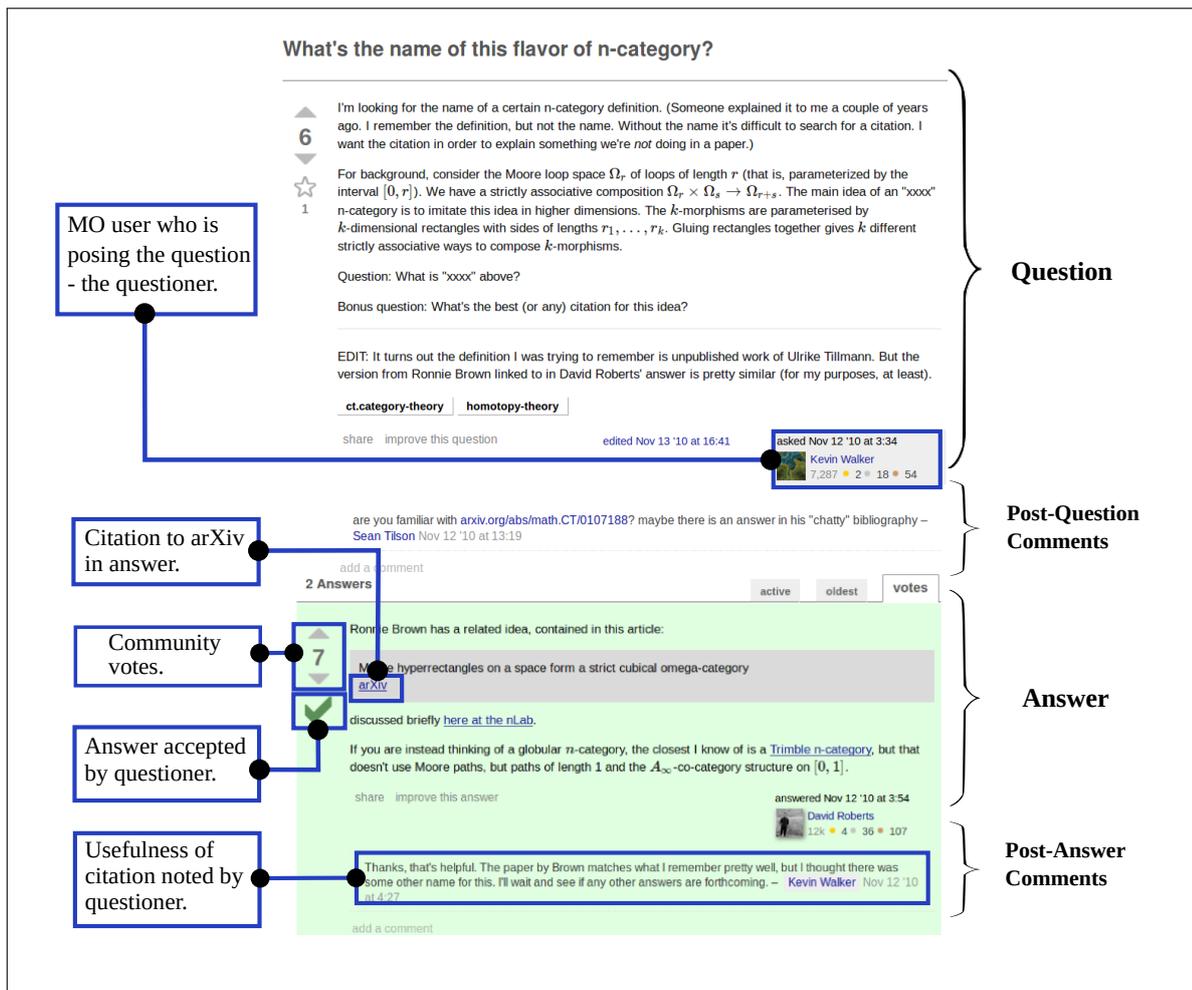


Figure 3.1: Anatomy of a MathOverflow thread (<http://mathoverflow.net/questions/45782>).

There are four parts: the question, the post-question (PQ) comments, the answer, and the post-answer (PA) comments. PQ comments are often used by community members to refine the question and make it more precise. PA comments are usually posted by the questioner to thank the answerer, acknowledge the usefulness of the answer and/or cited resource, or request clarifications from the answerer.

There are two mechanisms how the designated answer can be chosen. A questioner has the option to *accept* an answer posted in response to his or her question; in Figure 3.1, the green check mark indicates such a questioner-accepted answer. Acceptance of an answer does not automatically make an answer post the definitive response to a question; it is rather an indication that a particular answer post has been useful to questioner and satisfied his or her specific information need. Questioners are incentivised by MathOverflow to accept good answers by rewarding them with reputation points on the website<sup>5</sup>. Alternatively, the most highly voted answer post becomes the designated answer.

<sup>5</sup><https://mathoverflow.net/help/accepted-answer>

Several authors state that MathOverflow answers are authoritative, reliable and are produced by experts motivated to give helpful and relevant answers. We can see that best accepted answers tend to

- (a) cite papers that contain the answer in a self-contained manner;
- (b) summarise cited material; or
- (c) give an overview on how to obtain the answer.

(Tausczik and Pennebaker, 2011; Martin and Pease, 2013; Tausczik and Pennebaker, 2012; Tausczik et al., 2014). Martin and Pease (2013) studied the effectiveness of MathOverflow as a collaboration platform for producing mathematics by analysing qualitatively a sample of 100 MathOverflow discussions relating to group theory. The authors found that 90% of questions in their sample were successfully answered and that 56% of answers include citations to the research literature.

Another reason why it is reasonable to believe that answers on MO are reliable is that they are scrutinised quickly after being posted by a community of experts (Tausczik et al., 2014; Martin and Pease, 2013). Specifically, Martin and Pease (2013) observed that around 37% of the sampled answers had errors when initially posted by the answerers but that all errors were quickly pointed out by the MO community and corrected. Tausczik et al. (2014) found that primary (addition of new information) and secondary (revisions and extensions of existing material) contributions have an immediate impact on answer quality. In particular, acts such as providing information, revising and extending an answer, seem to be good predictors of immediate change in answer quality while indirect/evaluative contributions have a delayed impact on solution quality. Taken together, these observations suggest that the idea of procuring relevance judgements from MO answers might be feasible.

To summarise this far,

1. MathOverflow is primarily an academic community and its members view the website as a professional outlet for doing mathematics research.
2. MathOverflow users are often the most qualified experts to answer the questions posted on the site.
3. MathOverflow users are highly motivated to give good answers to the questions posted on the site.
4. Content on MathOverflow is heavily moderated and errors in questions and answers tend to be corrected quickly.
5. Documents cited in MathOverflow are relevant to the questions and likely to exhaustively answer them.

### 3.2.3 Queries and Relevance Judgements in an MO context

My plan is to use MO questions as information needs. To procure the relevance judgements for these information needs, observation on MathOverflow can similarly be used to provide these, in the form of academic citations in the attached accepted answer. If certain conditions are met, I will define the citations as positive relevance judgements on the cited papers. The use of citations in accepted MO answers as relevance judgements should be consistent with principles 1 (searcher is ultimate judge of relevance) and 3 (relevance is binary and constant) from section 2.4.3.

Harvesting citations from authoritative sources as relevance judgements – in the vein of the Cranfield II design (Section 2.4.1) – is an established practice in IR. Ritchie et al. (2006) elicited relevance judgements for citations in papers accepted in a scientific conference from their authors and used these judgements as part of their test collection of scientific publications. Fujii et al. (2006) constructed a test collection for the NTCIR-5 patent search task using expert reports for patents rejected by the Japanese Patent Office (JPO). The authors automatically extracted relevance judgements for their test collection by harvesting citations from the JPO reports used specifically to refute an applicant's patent demand. Fujii et al. (2006) argued that the harvested citations are authoritative relevance decisions because cited patents are identified by examiners of the JPO who are expert assessors of relevance. The same principle is applied by Graf and Azzopardi (2008) to cases from the European Patent Office (EPO).

**Ensuring Principle 4** My test collection should also ensure principle 4, the exhaustiveness of relevance judgements, but it is clear that it is not feasible to assess 439,000+ documents for relevance to the 100 or so queries I would most likely require. I therefore cannot *guarantee* that relevance judgements for my test collection are complete.

However, I claim that relevance judgements procured from MathOverflow using my proposed method are approximately complete, because of the fast response time to MO questions. I analysed 120 threads according to how long they have been exposed to experts on MathOverflow<sup>6</sup>, using the following 6 statistics:

1. **qcreated** and **acreated**. The number of days each **question** and **answer** respectively has been exposed to the MO community at the time the MO data dump was generated (20/10/2014).
2. **qedit\_creation** and **aedit\_creation**: The number of days from the date a **question** and **answer** respectively was created to the date it was last edited by the author of the question.
3. **qedited** and **aedited**. The number of days from the last edit of a **question** and **answer** respectively to the date the data dump was produced.

---

<sup>6</sup>These are the 120 threads selected for my test collection, as will be described in section 3.2.4.

Table 3.2 shows that refinement of questions on MO happens quickly, with a median time from edit to question creation being 0 (`qedit_creation` statistic, row 3 of Table 3.2). Questions used as topics in my test collection have been exposed to the MO community for a long period of time in their final edited form: the mean and median of the `qedited` distribution is 813.6 and 772.5 days respectively. I therefore assume the questions remained static because they are now of high quality (whether they were changed or not).

Scrutiny of answers on MathOverflow also happens quickly, as row 6 shows: a median of 0 for the `aedit_creation` statistic indicates that answers tend to be modified mainly during the first 24 hours of being posted on the site. Furthermore, answers from which relevance judgements for my test collection have been obtained have had long exposure since their final edit, with median and mean number of days since their last edit being 745.5 and 815.7 respectively.

|   | <b>Statistic</b>            | <b>Min.</b> | <b>1st Qu.</b> | <b>Median</b> | <b>Mean</b> | <b>3rd Qu.</b> | <b>Max.</b> | <b>Threads</b> |
|---|-----------------------------|-------------|----------------|---------------|-------------|----------------|-------------|----------------|
| 1 | <code>qcreated</code>       | 39.0        | 393.5          | 745.5         | 796.6       | 1228.2         | 1544.0      | 120            |
| 2 | <code>qedited</code>        | 10.0        | 342.5          | 772.5         | 813.6       | 1288.0         | 1544.0      | 68             |
| 3 | <code>qedit_creation</code> | 0.0         | 0.0            | 0.0           | 71.5        | 2.3            | 1442.0      | 68             |
| 4 | <code>acreated</code>       | 37.0        | 364.8          | 737.0         | 783.5       | 1220.0         | 1542.0      | 120            |
| 5 | <code>aedited</code>        | 66.0        | 430.0          | 745.5         | 815.7       | 1278.5         | 1542.0      | 58             |
| 6 | <code>aedit_creation</code> | 0.0         | 0.0            | 0.0           | 21.1        | 1.0            | 727.0       | 58             |

Table 3.2: CUMTC: Length of exposure of MO questions to experts on the site.

This means that the questions and answers used to produce my test collection have been exposed to a community of highly motivated experts for a relatively long period of time (median and mean number of days since their last edit being 745.5 and 815.7), while most questions are answered very quickly (most during the first 48 hours of being posted on the site). Given the amount of exposure the answers have had, it is reasonable to assume that missing relevant citations would have been pointed out quickly and harvested by my process (to be detailed in Section 3.2.4) from either the answers or the PA comments.

There are constraints on the questions of MO threads that will be included in my test collection. Questions should not be too broad or vague but rather express an information need that can be satisfied by describing objects or properties, stating conditions and/or producing examples or counter-examples. It is this property that makes an information need clear. The constraint on questions ensures that selected MO questions represent atomic information needs that can be satisfied by a self-contained answer involving concrete mathematical constructs: we are looking for questions with tangible answers.

MathOverflow questions represent real-life and research-level information needs because they arise from doing mathematics research (Tausczik et al., 2014). According to Tausczik et al., the mathematical problems underlying MO questions are (a) novel to the mathematician asking the question and (b) require original work but are solvable by domain experts. MO questions tend to be precise and unambiguous expressions of information needs because of moderation according to the house rules (Tausczik et al., 2014). This means that questions on MO are answerable,

|         |   |
|---------|---|
| Prelude | <p>There is a question someone (I'm hazy as to who) told me years ago. I found it fascinating for a time, but then I forgot about it, and I'm out of touch with any subsequent developments. It's a challenging question if I've gotten it right. Here it is:</p> <p>Suppose you have some kind of machine with two buttons, evidently designed by people with poor instinct for UI. The machine has many states in which the buttons do different things. Here are the assumptions:</p> <ul style="list-style-type: none"> <li>• There is no periodic quotient of the state space: no way to label states by an <math>n</math>-cycle so that both buttons advance the label by <math>1 \pmod n</math>.</li> <li>• It is not reversible: there are situations when two states merge into one.</li> <li>• It's ergodic: you can get from any state to any other state by some sequence of buttons.</li> </ul> <p>Now suppose its dinky little LCD is faded or broken, so you can't actually tell what the state it's in. (Formally, this is a finite state automaton, or an action of the free 2-generator semigroup on a finite set, and asks whether some element acts as a constant map).</p> |
| SQ-1    | Can anyone better identify the problem or fill in the history, and say whether it's still unsolved?   |
| SQ-2    | Is there necessarily a universal reset code, a sequence that will get you to a known state no matter where you start?   |

Table 3.3: MO post 38437, prelude and sub-questions.

real-life high-quality research-level information needs, and using them in my test collection should therefore satisfy Property 1 (Section 2.3.1).

Practically, I harvest the questions in the following way: The question body (text and formulae) of a MO thread can be considered as descriptions of information needs from which specific queries can be generated separately; following TREC tradition (Section 2.4.2), I therefore refer to MO questions as *topics*. Questions on MO often have multiple sub-questions, which I will refer to as *micro-topics* since they encode atomic information needs. I also observed that information in MO questions is typically carried by two types of sentences: *prelude* sentences, which are used to set the mathematical context (introduce mathematical constructs and results) and *query* sentences, which transcribe the information need itself and are in a discourse context with the accompanying prelude. For example, the prelude part and two sub-questions in MO question 146554 are shown in Table 3.3.

As far as the acceptable MO *answers* are concerned, they must contain at least one citation to an MREC document. The relationship between the cited resource and the answer can be classified along two dimensions:

- **Totality.** A cited resource is *total* if it contains all necessary information to derive the answer for the micro-topic or *partial* if it only addresses a special case. Citations that are total are

often the only resource cited in the answer and/or are explicitly identified as the resource that completely answers the question by the answerer, whereas citations that are partial are often accompanied by a range or condition.

- Directness. A cited resource is *direct* if the answer can be derived with little intellectual effort from its text, or *indirect* if the same information requires considerable effort (such as mathematical deduction or reasoning) for the questioner to reproduce.

I will now give an example of a direct and total citation.

**Question:** *I am looking for a reference which shows that the following statements are equivalent for a complex vector bundle  $E$ :*

- $E$  is a holomorphic vector bundle.
- There is a Dolbeault operator  $\bar{\partial}_E$ , i.e. a  $\mathbb{C}$  operator  $\bar{\partial} : \Omega^{0,0}(E) \rightarrow \Omega^{0,1}(E)$  which satisfies the Leibniz rule and  $\bar{\partial}_E^2 = 0$ .

*This is stated without proof in Huybrechts' Complex Geometry: An Introduction.*

**Answer:**

*A. Moroianu gives a detailed proof on pp. 72-74 of his Lectures on Kähler geometry (Theorem 9.2), available on the internet. (The preprint has it as Theorem 3.2.)...*

Looking into the actual MREC document, we can see that the solution is quite short:

THEOREM 3.2. A complex vector bundle  $E$  is holomorphic if and only if it has a holomorphic structure  $\bar{\partial}$ .

The answerer also uses the citation to outline the requested result and traces its origin to other authors.

**Ensuring Principle 2** While ideally all answers in my test collection should be total and direct, there are also cases of non-total and/or indirect answers. Principle 2 is the most challenging to verify because mathematical content often requires a synthesis of intermediate results through deduction and resolution. As a result, two or more papers that appear to be independently relevant (or not relevant at all) might be jointly relevant if results from both are synthesised in the answer. Here is an example of such a case, where the answer must be synthesized from material given by the answerer with more or less mathematical deduction<sup>7</sup>:

**Question:** *It is well known that if  $c(K) = 2n + 1$ , then  $u(K)$  is less than  $n + 1$ . It can not be sharper because of the trefoil knot. On the other hand, if  $c(K) = 2n$ , then similarly we have  $u(K)$  is less than  $n + 1$ . I think  $u(K) = n$  is impossible in this case, i.e. there does not exist a knot  $K$  with  $c(K) = 2n$  and  $u(K) = n$ . Maybe it is fairly easy, but I have no idea how to deduce it. Any hint is welcome :).*

---

<sup>7</sup>MathOverflow thread 109218

In this case the answerer responds with a direct citation:

*You can see the answer in Proposition 2.1 of link text .*

The questioner then documents the reasoning he or she has performed, using the material in the cited document, in the PA comments to construct the answer:

*Thank you very much for your response. Given a knot with  $c(K) = 2n$ , then  $u(K) \leq a(K) \leq n - 1$ . Although  $a(K)$  (the ascending number) is not easy to calculate in general, it works well in this question.*

The test collection constructor must follow the argumentation for the right answer during this process. My test collection construction algorithm (in particular the determination of totality and directness) therefore carefully documents these argumentation steps so that principle 2 problems can be addressed. The role of citations in this argumentation is an important aspect: the clearer the argumentation for the right answer, the simpler the process of determining the role of cited resources.

**Citations and Inference** Let us now look at the roles a citation can play in answering a particular sub-question:

1. **Complete:** The cited resource gives the full answer to the question.
2. **Component:** The cited resource is used to import components (results such as theorems, lemmas etc.) that are necessary to compose the answer (e.g., through inference). This can be acceptable, if all necessary components for a proof are present.
3. **General:** The cited resource contains a more general result than the one asked for by the questioner.

It is not uncommon for answerers to cite results that are more general than what is needed to resolve the question. For example, in response to the question<sup>8</sup>:

*Let  $A_n = \{a \cdot b : a, b \in \mathbb{N}, a, b \leq n\}$ . Are there any estimates for  $|A_n|$ ? Will it be  $o(n^2)$ ?*

the answerer responds:

*This question is known as the multiplication table problem, and was originally posed by Erdős in 1955. Erdős proved that  $|A_n| = o(n^2)$ , and this was sharpened by Tenenbaum in 1984. In 2008, Ford gave the exact magnitude and proved that*

$$|\{a \cdot b : b \leq N\}| \asymp \frac{N^2}{(\log N)^c (\log \log N)^{3/2}}$$

---

<sup>8</sup>MathOverflow thread 108912

where

$$c = \frac{(1 + \log \log 2)}{\log 2}$$

In 2010 Koukoulopoulos gave multidimensional generalizations of Ford's result, proving that

$$|\{a_1 \dots a_{k+1} : a_i \leq N \text{ for all } i\}| \asymp \frac{N^{k+1}}{(\log N)^{c_k} (\log \log N)^{3/2}}$$

, where

$$c_k = \int_1^{\frac{k}{\log(k+1)}} \log x dx \frac{\log(k+1) + k \log(k) - k \log \log(k+1) - k}{\log(k+1)}$$

Sometimes, a partial or more general result is the only result available to a particular mathematical problem. Therefore, if an MREC citation addresses a more general form of the questioner's problem in my test collection, I will accept it, provided the general result is self-contained in the cited document.

4. **Example:** The role of the cited resource is to point to examples and counter-examples. This will also be accepted, if the example was what the questioner asked for.

Apart from these 4 useful roles, there are also 5 different ways how a citation can be only marginally related to answering the question:

1. **Informational:** The cited resource is used by the answerer to point to more information that can provide details to related material. Reading the resource, however, is not necessary for constructing the answer and informational citations are not useful as relevance judgements.
2. **Starting point:** The cited resource is included as a starting point from which the questioner can start looking for the answer. Starting point citations are not useful as relevance judgements because considerable intellectual effort is required to derive the answer from their content.
3. **Background:** The cited resource is used by the answerer to direct the questioner to background material and is not directly relevant to the argumentation in the answer. As a result, citations to background material are not useful as relevance judgements.
4. **Irrelevant:** The citation is irrelevant to the question. I label a citation as "irrelevant" if the questioner explicitly characterises the cited document as such in the PA comments section.
5. **Unclear:** The role of the citation is not clear from the text of the answer. I assign this role when the role of a citation in forming the answer is unclear, or by *exclusion*: nothing else best describes their role. Citations whose usefulness is unclear, with respect to obtaining the answer, are not included in my test collection as relevance judgements.

After completing this discussion of factors playing into the process of selecting suitable MO questions and MO answers with citations, I will now describe the practical algorithm for creating the MO test collection.

### 3.2.4 The Construction Process

#### Data Preparation

Any MO data dump<sup>9</sup> encodes mathematical expressions in posts as embedded  $\LaTeX$  code –  $\$<formula\ code>\$$  for inline expressions and  $\$\$<formula\ code>\$\$$  for formulae intended for display on a separate line. I used the `LatexMathML` tool to convert  $\LaTeX$  into MathML in order to ensure maximal compatibility with the MREC: this is the same tool used by the creators of the ArXMLiv and MREC corpora for the same conversion (Stamerjohanns and Kohlhase, 2008; Stamerjohanns and Kohlhase, 2009; Stamerjohanns et al., 2010; Liška et al., 2011).

The first step in my construction process is to identify MO threads whose accepted answers cite at least one MREC document (but may also include additional citations external to the MREC).

I used MREC meta-data, file paths and arXiv document IDs to compile a list of over half a million simple string patterns that reflect how a document might appear within a URL to arXiv.org. For example, the MREC document with arXiv ID “0910.0885” (Gauld, 2009) might be cited in an MO answer using a URL to its pdf, or to the arXiv.org page for that document. Thus, my program generates one string pattern for each method:

(a) “`https://arxiv.org/pdf/0910.0885.pdf`”;

(b) “`https://arxiv.org/abs/0910.0885`”.

I sampled MO threads with citations in their answers using the following string matching signals:

- match specific terms such as “reference”, “check this”, “read”, “found here”, “article”, “paper”;
- match URLs pointing to pdf, doc or ps documents;
- match references to years after 1800, such as 1985 and 2001;
- match page number references, such as “p. 27–39”, “page 27-39”, “27–39”;
- match references to volume and number such as “191 (2012)”, “191:2”, “no. 2”, “number 2”.

---

<sup>9</sup>I used the MO data dump of 20/01/2014.

The list of simple patterns is input to a finite-state transducer (FST) that decides whether a candidate URL in an MO answer contains one of the patterns in the list. My FST is implemented using the Aho-Corasick algorithm multi-pattern string matching algorithm (Aho and Corasick, 1975). The list of patterns is compiled into an Aho-Corasick automaton that can match all patterns simultaneously in one pass of the input text. Matching  $m$  patterns against a candidate URL of length  $n$  characters takes  $O(n + m)$ .

Finding citations to MREC documents by looking at URLs in MO answers is prone to producing false negatives. For example, answerers might cite MREC documents using their title, rather than URLs to arXiv.org. My approach is a compromise induced by the scale of the problem. Out of a total of 47,753 discussion threads, my method has identified 357 as having accepted answers citing at least one MREC document. I will refer to this set of 357 threads as the “seed set”.

### **Manual Assessment**

In the manual assessment phase, I further filtered the 357 threads found in the previous phase. First, I identified individual sub-questions in MO questions. This is a necessary refinement since sub-questions correspond to atomic information needs and correspond to distinct micro-topics in my test collection. Second, I inspected the answers and assigned roles to each citation, with respect to every sub-question in the thread.

Although it is guaranteed that each answer in the seed set contains at least one citation to the MREC, it might also contain additional citations to external resources. But if a sub-questions relies solely or partially on citations external to the MREC in order to be answered, the sub-question must be excluded from the test collection. Third, I evaluated the suitability of sub-question/MREC citation pairs with respect to the ideal requirements above. Pairs that meet both criteria are included in my test collection as micro-topics and relevance judgements. This is done in the following 8 steps:

**Step 1.** I manually segmented the question part of each MO thread into sentences, and distinguished those that express a question (as “query” sentences) from those that introduce the mathematical context (as “prelude” sentences).

**Step 2.** I grouped “query” sentences into separate sub-questions, each group representing an atomic information need. To do so, I needed to determine whether a “query” sentence is independently related to the “prelude” sentences, or if it is bound to sub-questions expressed in other “query” sentences in the thread. Finally, I assigned a unique identifier to each micro-topic (i.e., a sub-question and all “query” and “prelude” sentences related to it). (This is often trivial as questioners tend to itemise or explicitly number their sub-questions.)

**Step 3.** I examined post-question (PQ) comments for question answerer’s acceptance of answers. In some occasions, the answer is first posted in this part of the discussion, which could

also mean that the questioner's confirmation of the usefulness of the answer might also be found there.

**Step 4.** I constructed a step-by-step derivation of the answer for each sub-question in the thread. If MO answers are complex, contain a pointer to the answer instead of an explanation and/or the explanation contains citations to both MREC and non-MREC documents, this is a prerequisite for steps 5 and 6. Complex MO answers are those that (a) are long, (b) are broken-up in many sub-cases/steps, or (c) involve some calculation and/or make use of notation that is specific to the underlying mathematical domain.

When constructing the derivation, I followed mathematical argumentation from both the answer and the MREC document cited therein using cues from the text. Set phrases in the answer text, such as "it follows" and argument patterns such as "since X, we have Y" are often good signals of mathematical inference. For each step in the mathematical derivation of the answer, I copied its sentences ad verbum on a clean sheet of paper. This process resulted in a numbered sequence of steps that answer the question.

**Step 5.** For each subquestion, I identify the relevant parts of the answer from Step 4. (This task can be trivial in cases where the questioner has enumerated sub-questions explicitly.) In most cases I found that this mapping can be done in an objective manner by following the argumentation and observing the names of the mathematical objects and structures involved.

**Step 6.** For each sub-question, I considered all MREC citations from the answer associated with it (steps 4 and 5), and assessed each citation's role and relevance to the sub-question. I used the 4 useful and 5 less useful labels from page 70 to do so.

**Step 7.** In this step I decided whether every sub-question/MREC citation pair in the MO question will be part of my test collection or not. Making this decision involves two sub-steps. First, I test whether the sub-question is clear and its answer produces a concrete example, mathematical object or structure. If it isn't, the sub-question/MREC citation pair is rejected.

Second, I used information from steps 4, 5 and 6 to decide whether an MREC citation is useful as a relevance judgement. Pairs with citations that are assigned the "background", "informational", "starting point", "unclear" and "irrelevant" roles are excluded from my test collection because they are in violation of Principle 1.

For each remaining candidate pair, I used information from steps 4, 5 and 6 to decide whether the pair was total, and whether it was direct. The step-by-step reconstruction of the argumentation in the answer (step 4) is useful in two ways. First, it enables me to identify whether a citation was used to import only a small part of the answer (e.g., a "component"). In this case, the pair is in violation of criterion 2 and is rejected. Second, it allows me to quantify the amount of deduction needed to derive the answer from the cited documents: an MREC document that answers the sub-question using some deduction and/or resolution is acceptable provided it does

so in a self-contained manner (e.g., only uses theorems developed in its body). All non-direct pairs are immediately rejected.

Pairs that are total or partial on the totality axis and direct on the directness axis satisfy criterion 2 of the ideal standard and are included in my test collection. I accept pairs classified as “partial” on the totality axis because their MREC citations are relevant to the sub-questions, despite the fact they only address a special case.

**Step 8.** I read through the PA comments and looked for confirmation of the usefulness of the answer/citations by the questioner (most acceptances are found during this step). Sometimes, I also found expressions of doubt by the question answerer in the PA comments, which might lead to the exclusion of subquestion/citation pairs. Additionally, I looked for additional MREC citations in the PA comments supplied by either the answerer or the questioner. Such additional MREC citations posted by 3rd-parties can also be included in my test collection provided that their usefulness is confirmed by the questioner in the PA comments.

This process is reasonably objective because it delegates all domain-specific decisions about relevance to experts on the website; the process is systematic and requires little expertise in mathematics for anybody to apply it. (My supervisor and myself independently applied my process on 6 example MO threads and agreed on the classification of relevance judgements in 5 of these cases.) In all cases, the questioner remains the ultimate judge of relevance (by either accepting an answer directly or by explicitly commenting on the relevance of posted material, as recorded in steps 3 and 8 of the process).

At the end of this process, 120 threads covering 160 micro-topics have been determined to be compatible with the ideal-standard criteria and have been accepted for inclusion in my test collection. Examples of the application of my process to MathOverflow threads are discussed in Appendix B.1.

### 3.2.5 Test Collection Overview and Statistics

The completed CUMTC contains 160 micro-topics, organised in 120 topics (MO threads). It contains 184 positive relevance judgements involving 224 unique MREC documents; with all other MREC documents being declared as irrelevant by definition.

| <b>Micro-topics</b> | <b>1</b> | <b>2</b> | <b>3</b> | <b>≥ 4</b> | <b>Total</b> |
|---------------------|----------|----------|----------|------------|--------------|
| Instances (topics)  | 88       | 24       | 8        | 0          | 120          |
| Percentage (topics) | 73.3%    | 20.0 %   | 6.7%     | 0%         |              |

Table 3.4: Topic/Micro-topic break-down

Table 3.4 is a break-down of topics by the number of micro-topics they encapsulate. The vast majority of topics (93.33%) have either 1 or 2 micro-topics, with the average being close to 1 (1.33).

| <b>Relevant documents</b> | <b>1</b> | <b>2</b> | <b>3</b> | <b>4</b> | <b>Total</b> |
|---------------------------|----------|----------|----------|----------|--------------|
| Instances (topics)        | 97       | 21       | 1        | 1        | 120          |
| Percentage (topics)       | 80.83%   | 17.5%    | 0.83%    | 0.83%    |              |
| Instances (micro-topics)  | 140      | 17       | 2        | 1        | 160          |
| Percentage (micro-topics) | 87.50%   | 10.625%  | 1.25%    | 0.625%   |              |

Table 3.5: Number of relevance judgements per topic (above) and micro-topic (below).

Table 3.5 shows that the majority of topics (80.83%) have only one relevant document while a further 21 (17.5%) have two relevant documents. Two topics have more than 2 relevant documents: one with 3 and another with 4. In terms of micro-topics, this corresponds to 140 micro-topics (87.50%) with 1 relevant document, 17 (10.625%) with 2, 2 micro-topics (1.25%) with 3 and just one (0.625%) with 4 relevant documents. Only 12.5% of micro-topics have more than one relevant document.

As shown in Table 3.6, 106 topics (88.33%) contain a mixture of text and formulae while 14 (11.66%) topics only contain text. In terms of micro-topics, 116 (72.5%) contain both text and formulae in their body while the remaining 44 (27.5%) only contain text.

|                           | <b>Formulae + text</b> | <b>Text only</b> | <b>Total</b> |
|---------------------------|------------------------|------------------|--------------|
| Instances (topics)        | 106                    | 14               | 120          |
| Percentage (topics)       | 88.33%                 | 11.66%           |              |
| Instances (micro-topics)  | 116                    | 44               | 160          |
| Percentage (micro-topics) | 72.50%                 | 27.50%           |              |

Table 3.6: Number and percentage of topics and micro-topics with and without Formulae.

| <b>Distribution</b>    | <b>Min.</b> | <b>1st Qu.</b> | <b>Median</b> | <b>Mean</b> | <b>3rd Qu.</b> | <b>Max.</b> |
|------------------------|-------------|----------------|---------------|-------------|----------------|-------------|
| Words                  | 4.0         | 46.5           | 97.0          | 107.3       | 153.5          | 371.0       |
| Formulae               | 0.0         | 0.0            | 5.0           | 7.24        | 10.0           | 55.0        |
| All tokens             | 4.0         | 52.5           | 102.5         | 114.6       | 157.0          | 426.0       |
| Formula-to-Token ratio | 0.0         | 0.0            | 0.060         | 0.062       | 0.097          | 0.214       |

Table 3.7: Statistical distribution of tokens (formulae and words) in 160 micro-topics.

Table 3.7 shows the distribution of tokens (formulae and words) that make up the micro-topics in my test collection. On average, there are 107 words in the micro-topics, with a maximum of 371 words. In contrast, on average there are about 7 formulae per micro-topic, with a maximum of 55 formulae. The formula-to-token ratio, that is the proportion of formulae over all tokens, averages at 0.062 with a maximum of 0.214, as shown in the bottom row of Table 3.7.

**Development Set** I also had to compile a development set that is distinct from the main collection. However, sometimes it is necessary to observe how changes in a retrieval model affect performance, carry out model tuning and assess retrieval efficiency under different configurations. Being reserved for evaluation, data in my test collection cannot be used for this purpose. In this thesis, the development set is necessary to objectively approximate the  $\alpha$  parameter for

the Tangent system (Section 2.2.5), for example. Topics and relevance judgements in the test collection are expensive to procure – they are sparse in the data and considerable effort is required to evaluate their suitability. My test collection is intended to be used as a standardised evaluation set enabling evaluation on unseen data.

Ideally, the development set is composed of material that is comparable to that in the main collection. Thus, one way of producing a development set is to reserve a small subset of the main test collection for development. However, I chose not to follow this approach for the following reasons:

1. I opted to keep my evaluation set as large as possible in order to increase confidence in the statistical soundness of my evaluation.
2. Since a development set is not intended to be used for formal evaluation, the criteria for selecting MO threads for development can be relaxed. For example, during the compilation of the main test collection, I laboriously attempted to establish the connection between topic and document as well as the completeness of the relevance judgements with high confidence. Moving data-points of this quality to a development set would be wasteful.

Instead I used those MO threads that were not included in the main test collection for “borderline” reasons. The resulting development set is small, a total of 13 micro-topics, 8 of which contain both formulae and text.

### 3.3 Chapter Summary

This chapter presented a methodology for constructing test collections composed of real-life, research-level mathematical information needs. Information needs and relevance judgements are procured by observing questions and answers on the MathOverflow online website, a community of expert research mathematicians.

My method generates reasonably objective and verifiable relevance judgements and it can be applied without detailed knowledge of the domain matter. The end product of my process, the Cambridge University Mathematics Test Collection (CUMTC) is a test collection that is:

- realistic, because it models the operational setting of searching for research-level mathematics: (a) the topics in my collection are information needs from real research mathematicians (section 3.2) and (b) the documents in my test collection are representative of the kind of documents research mathematicians search for (section 3.2.1).
- sufficiently consistent with principle 1 in section 2.4.3, i.e., they are derived from real users with relevance judgements by the same user. MO questions in my test collection have answers that have been implicitly approved by the authors of the corresponding questions. As we have seen in sections 3.2.2, MO questions tend to be answered not long after being posted on the site.

- sufficiently consistent with principle 2 (independence of relevance judgements) in section 2.4.3. My process for selecting micro-topics explicitly addresses the problem of relevant mathematical results being distributed in multiple, interdependent documents (section 3.2.4).
- consistent with principle 3 of section 2.4.3, because relevance judgements are binary relevance judgements and constant (the test collection is static).
- sufficiently consistent with principle 4 (completeness) of section 2.4.3, and thus reusable, because of the prolonged exposure to a very reactive platform. On this platform, experts respond quickly, which ensures that all known relevant documents are identified. In a context like research-level mathematics, there are by definition few relevant documents per question (most likely one for most questions).
- composed of high quality relevance judgements, because they have been produced by authoritative domain experts (section 3.2.2). As demonstrated by Bailey et al. (2008), experts produce relevance judgements that are of higher quality than those produced by non-experts.

In the next chapter I will introduce mathematical types in detail, describe two algorithms for automatic type extraction from large mathematical corpora, and present classical, text-based retrieval models with type-aware extensions. Chapter 5 will build on the idea of types by devising a machine learning task for assigning types to variables as denotations. In chapter 6, I will develop typed retrieval models that exploit the link between variables and types. I will use my test collection CUMTC to evaluate my text-based type-aware and typed retrieval models in chapter 7.

# Chapter 4

## Mathematical Types for Retrieval

Mathematical types are sequences of one or more words used to label mathematical objects (e.g., “set”, “smooth curve”), algebraic and non-algebraic structures (e.g., “graph”, “monoid”, “group”) and mathematical notions (e.g., “cardinality of a set”). Mathematicians adopt consistent labels for concepts in order to evoke them in the learned discourse without having to redefine them. The collective of mathematicians decides which concepts are important enough to be named and what labels are used to evoke them. Gradually, labels for well-understood concepts in each specialisation enter the mathematical lexicon in predictable ways. As named mathematical concepts shared between mathematicians, types play a central role in communicating mathematical information: they enable mathematicians to manipulate mathematical concepts, assign properties to objects and prove assertions about them. Types are central to my investigation, and the hypotheses in this thesis stem from the above observations about types.

In this chapter, I will develop the infrastructure needed to perform MIR experiments based on textual types, and lay the foundations needed for joint text and formula retrieval based on types. I will describe methods for automatic type detection in section 4.2. Since types in query text are subject to sparseness, I will explain how query expansion can be applied using query types to match similar but not identical types in documents (section 4.3). In section 4.4, I will explain the pipeline I built to perform practical textual typed retrieval. I will extend existing bag-of-words retrieval models to perform text-based retrieval based on types as atomic tokens, rather than multi-word phrases. At the end of this chapter, this puts me in a position to empirically test my first hypothesis, which posits that research level math IR will benefit from treating mathematical types in text as atomic units.

A little preview into chapters 5 and 6 before I get into the topic of this chapter: my second hypothesis is that joint modelling of textual and symbolic information is beneficial to retrieval effectiveness. I will use type information to connect the two. There are two modalities of types: the explicit, textual one and the symbolic one which is implicit, i.e., has to be inferred. Types naturally mediate between the textual and symbolic contexts: mathematically, they are associated with variables and thus the formulae, and linguistically, they are explicitly described in text. This

observation will eventually allow me to develop MIR models where the equivalence classes used in both indexing and scoring have access to more information, e.g., can perform a simple form of type inference. On the way to get there, chapter 5 will provide a method to perform variable typing automatically, and chapter 6 will exploit the type link for practical joint text and formulae MIR.

I now turn to notation:

**Definition** Typed variable notation. The notation

$$v_1, \dots, v_n :: T$$

means variables  $v_1, \dots, v_n$  have been assigned type  $T$ , or equivalently:

$$v_1 \text{ is typed } T, v_2 \text{ is typed } T, \dots, v_n \text{ is typed } T.$$

**Definition** Inference notation. I will use the following notation to define rules of shallow type inference :

$$\text{rule name} \frac{\text{premise}_1 \wedge \dots \wedge \text{premise}_n}{\text{conclusion}}$$

Let us now return to the notion of  $\alpha$  equivalence (introduced on page 27). Annotating Symbol Layout Trees (SLTs, Section 2.2.1) with type denotations for variables can be useful in identifying semantically distinct but syntactically equivalent ( $\alpha$ -equivalent) mathematical expressions. Consider, for example, the  $\alpha$ -equivalent expressions (1)  $a + b$  and (2)  $x + y$ . Suppose that  $a, b :: \text{NUMBER}$  and  $x, y :: \text{VECTOR}$ . From a lexical point of view, the expressions (1) and (2) are indistinguishable as they share the same structure  $\text{Expr}_1 + \text{Expr}_2$ . In contrast, with typing applied, the two expressions can be shown to have semantic distinction since (1) is semantically addition of numbers  $\text{NUMBER}_1 + \text{NUMBER}_2$ , while (2) represents addition of vectors:  $\text{VECTOR}_1 + \text{VECTOR}_2$ . Instances of this distinction may result in an improvement in precision due to a reduction to the number of false positives.

Conversely, expressions that contain different lexical tokens may be semantically equivalent. In the context of the earlier example, suppose that  $a, b, x, y :: \text{NUMBER}$ . Given this type information, an intelligent MIR system would be able to deduce that formulae (1) and (2) are identical, despite their lexical differences. Retrieval based on typing mathematical expressions using types in the text should be able to identify such instances and improve recall. I will introduce and evaluate models based on this approach in chapters 6 and 7 respectively.

Types are technical terms and are thus implicitly organised, through compositionality, in a hierarchy of concepts (section 4.1). For example, a ‘‘Complete Metric Space’’, is a specialisation of ‘‘Metric Space’’, which in turn is a specialisation of ‘‘Space’’.

In my approach to typed retrieval, I exploit compositionality of types to make inferences about the relatedness of types and typed variables via *type-casting*: If type  $T_1$  is a subtype of type  $T$ , then  $T_1$  can be considered, for the purpose of typed indexing and retrieval, an instance of  $T$ . This implies that any inference that can be applied to  $T$  can also be applied to its subtypes:

$$\text{type-cast} \frac{T_1 \rightarrow_t T}{\text{inf}(T) \Longrightarrow \text{inf}(T_1)}$$

where  $\rightarrow_t$  is the transitive reflexive closure of the subtype relation, and  $\text{inf}$  is any type inference operation. In the context of this thesis, type casting is an informal and on-the-surface form of type coercion introduced by Ganesalingam (2008). The type coercion relation generalises the ideas of subtyping and type casting in a formal type system.

I use three informal type inference rules to build an off-line dictionary of types and to perform typed indexing and retrieval:

1. **Type detection:** If I know that  $T$  is a type and  $T_1$  is a longer phrase with  $T$  as its suffix, then  $T_1$  is also a type:

$$\text{type-intro} \frac{\text{istype}(T) \wedge T_1 \rightarrow_s T}{\text{istype}(T_1)}$$

where  $A \rightarrow_s B$  is the suffix relation:

“**B** is a suffix of **A** of arbitrary length up to and including **B**”.

This rule is therefore called the `type-intro` rule.

2. **Type disambiguation** (Section 6.3.1). Suppose that a variable  $x$  is assigned two different types at different points in the mathematical discourse, say using the following sentences:

Let  $x$  be a **Borel subgroup**.

Let  $x$  be a **parabolic subgroup**.

In typed retrieval a type-aware indexer must decide which of the two candidate types (“Borel subgroup” and “parabolic subgroup”) to assign to  $x$ . In section 6.3.1 I propose the use of a *type disambiguation algorithm*. One component of my algorithm is the following rule:

$$\text{type-disamb1} \frac{T_1 \rightarrow_t T \wedge \dots T_n \rightarrow_t T \wedge x :: T_1 \wedge \dots x :: T_m \wedge 1 \leq m \leq n}{x :: T}$$

In the context of the example above, the application of this rule would determine that the most general type for  $x$  (i.e., the most general unifier type) is the supertype of both candidate types: “subgroup”.

3. **Type unification:** Two variables  $a :: T_1$  and  $x :: T$ , coming from different formulae, are type-unifiable if the type of one ( $T_1$ ) is a subtype of the other ( $T$ ):

$$\text{type-unif} \frac{a :: T_1 \wedge x :: T \wedge T_1 \rightarrow_t T}{\text{of sametype}(a, x)}$$

In section 6.4.1 I will describe typed retrieval models based on tree matching that reward structurally identical query–document formulae pairs that are also type-unifiable. The type embedding space complements the `type-disamb1` and `type-unif` rules above in my implementation of type disambiguation and unification (sections 6.3.1 and 6.2 respectively). In both implementations, I will use the type embedding space to discover new types that are related to both candidates when the candidate types have no common supertype. In text-based retrieval, I will use the type embedding space to expand queries in the CUMTC with related concepts.

In the next section I will present a detailed definition of mathematical types and discuss the linguistic properties of types, as a special case of technical terms. Subsequently, in section 4.2 I will present two algorithms for automatic type detection. I apply my algorithms to extract a type dictionary from the MREC, but my methods can be applied on any large mathematical corpus. I discuss how I built my type embeddings space in section 4.3. In section 4.4, I introduce type-based text retrieval of research-level mathematics in order to investigate the first hypothesis of this thesis. In section 4.5, my approach to modelling mathematical concepts using types will be compared to other models proposed in the MIR literature.

## 4.1 Definition of Mathematical Types

There is a close relationship between my definition of types and technical terms – mathematical types are a subset of technical terms. The structure and statistical properties of technical terminology are generally well understood. Most technical terms are noun phrases, but in some rare cases, technical terms might also contain verbs, adverbs or conjunctions (Justeson and Katz, 1995). Justeson and Katz defined technical terms as a special instance of noun phrases called *lexical noun phrases* – noun phrases whose meaning cannot be unambiguously determined by the meanings of the words that compose them. As a result, technical terms have to be repeated as full lexical noun phrases in the discourse – omitting modifiers from a technical term (e.g., for abbreviative purposes) will result in a reference to a different concept than the one intended (Justeson and Katz, 1995). As technical terms are lexical noun phrases that are repeated in their entirety in the text, they can be automatically detected in a large corpus. One of the most commonly used algorithms for this is the C-Value algorithm (Frantzi et al., 1998). This algorithm uses corpus-wide frequencies of a potential term, its subsequences and supersequences<sup>1</sup>, and combines these with information gained from the contexts of other potential terms. It also applies linguistic filters in the form of a stop-word list and regular expressions over part-of-speech sequences<sup>2</sup>.

Gödert (2012) identifies two additional characteristics of mathematical technical terminology. First, the majority of concepts in mathematics can be described by technical terms that are compounds of adjectives and nouns. Second, many mathematical technical terms are eponyms,

---

<sup>1</sup>The sequences that contain the potential term as subsequences.

<sup>2</sup>Details of the algorithm can be found in the appendix A.1.5.

i.e., named after their inventors with additional pre and post modifications (e.g., “refined Noether normalization theorem”, “abstract Hilbert space theorem”).

A type is any technical term that is (a) perceived by mathematicians to refer to mathematical objects, algebraic structures and mathematical notions and (b) can be instantiated in the discourse in the form of a variable. While this notion of a type is intuitive to most mathematicians (as I will demonstrate in section 4.2.2), it is nevertheless hard to produce a concrete list of properties a technical term must adhere to in order to be considered a type. As a result, I will take a constructive approach to defining types: I will describe which technical terms can be considered to be types, give examples and identify exceptions. All other technical terms can be considered to be non-types.

A mathematical object is anything that can be formally defined and manipulated in the discourse as part of formal deductive reasoning and/or proofs. Some objects such as “Number”, “Matrix” and “Set” are considered basic and are never explicitly defined by mathematicians (Ganesalingam, 2008).

Mathematical structures (including algebraic ones), like objects, take part in mathematical manipulation but are defined as collections (or tuples) of other objects. As an example, consider the following definition taken from Diestel (2012):

A **graph** is a **pair**  $G = (V, E)$  of *sets* satisfying  $E \subseteq [V]^2$ ; thus, the *elements* of  $E$  are 2-element *subsets* of  $V$ . To avoid notational ambiguities, we shall always assume tacitly that  $V \cap E = \emptyset$ . The *elements* of  $V$  are the **vertices** (or **nodes**, **points**) of the **graph**  $G$ , the *elements* of  $E$  its **edges** (or *lines*).

where the words in bold are technical terms that refer to mathematical structures and words in italic are references to mathematical objects. Both of these are types.

Mathematical notions are abstract mathematical concepts that can be instantiated as variables or can take the form of other objects. For example, an “envelope of elliptic trajectories” can be a “parabola” (Richard, 2004) or an “ellipse”<sup>3</sup>.

Named axioms, theorems and conjectures are also considered to be types since they refer to universally accepted, formally defined constructs for the purpose of argumentation (e.g., to complete a proof). Named results have an additional utility in information retrieval; they possess discriminating power because they often refer to concepts that uniquely identify the related topic. For example, the “initial value theorem” is distinctively associated with topics in mathematical analysis.

Types are part of the mathematical terminology, but not every technical term is a type. Examples of non-types include:

- references to mathematical procedures (e.g., “proof by contradiction”);

---

<sup>3</sup>See answer in MathOverflow question 30402 <http://mathoverflow.net/questions/30402/parabolic-envelope-of-fireworks>.

- elements of the mathematical discourse (e.g., “Theorem 4.1”, “left-hand-side”);
- characterisations of mathematical notions (e.g., “statistically significant”);
- properties of operators (e.g., “Associativity”);
- mathematical processes (e.g., “Differentiation”) and
- theories (such as “Chaos”). Note that mentions of mathematical theories are ambiguous: it is often unclear whether they refer to a branch of mathematics or to a formally defined construct.

These examples are not types because either

- (a) they are not explicitly referring to mathematical constructions (i.e., they cannot assign meaning to variables);
- (b) their role in the discourse is indirect (e.g., properties capture relationships between concepts, see below) and
- (c) they are used as discourse labels for anaphoric purposes (e.g., “Theorem 4.1”).

I will now explain further why I do not treat *properties* as types in this thesis. First, properties as technical terms fail the instantiation test – they cannot be instantiated or manipulated in the discourse as variables. For example, mathematicians would write “Let  $x$  be an *associative operator*” but not “Let  $x$  be associativity”. Second, properties model the interaction between objects whereby one object modifies another. For example, consider the following sentences:

Let  $\oplus$  be a binary operator over  $R$ .

Suppose that  $\oplus$  is associative.

In the first sentence the symbol  $\oplus$  is introduced as a binary operator in the discourse. This binary operator is subsequently modified further to become an *associative binary operator* in the second sentence. Operationally, the second sentence updates the context such that the equation underpinning the associativity axiom (the modifier object) is presumed to be true for the object introduced in the first sentence and denoted by  $\oplus$  (the object being modified). This interaction is distinct from the one I intend to model in this thesis, namely the assignment of types from the textual modality as denotations to variables in the symbolic modality.

Gödert (2012) notes that a unique characteristic of mathematical terminology is that it re-purposes words from everyday life, such as “field”, “chain” and “root”. In mathematics, these words assume meaning that is different from that of everyday use. This accounts for the fact that types display a high level of polysemy (“field” in Mathematics is a concept distinct to that in Physics) and synonymy (e.g., “karoubi envelope” is the same as “category of idempotent arrows”).

New types can be formed by modifying other types in three ways. The first way is through parameterisation. For example “2-Group” and “ $G$ -Function” are parameterisations of “Group” and “Function” respectively.

The second way was also observed by Katz for all technical terminology: that due to the structure of technical terms, they are organised as taxonomies, with longer technical terms being special instances of smaller, more general concepts. This relationship is often mirrored linguistically: a type expressed by a longer string is often a subtype of the types in its suffixes. Compositionality between types is a consequence of how technical terms in mathematics are formed. So, a second way to form new types is by pre-modifying shorter types with adjectives and nouns. For example, a “Borel subgroup” is a specialisation of the type “subgroup”, the former obtained by pre-modifying the latter with the proper noun “Borel”<sup>4</sup>. A third way to form new types is by post-modifying shorter types with prepositional phrases. For example, “Set of Numbers”, “Ideal of a Ring”, “Point on the Plane” and “Set of Matrices” are new concepts based on “Set”, “Ideal” and “Point” respectively.

Not all type/sub-type relationships are expressed on the surface. For example, it is not obvious that “Klein Bottle” is a sub-type of “Surface”. As we will see in sections 4.3 and 7.1.2, modelling the meaning of type labels as type vector embeddings in a type embedding space can serve to discover topically similar types. This observation can be used to approximate non-lexical super/sub-type relationships for the purpose of *type disambiguation* (section 6.3.1).

## 4.2 Automatic Type Detection

In order for types to be useful in practice, type phrases must be detectable and extractable at scale. For this purpose, I developed a type detection algorithm. The algorithm’s first stage is based on the intuition that any technical term that is a type will have a corresponding entry in a mathematical reference resource, such as the Encyclopedia of Mathematics<sup>5</sup>. The first stage therefore constructs a “seed dictionary” of types. The second stage, which I call the “double suffix-trie algorithm”, takes two inputs: the seed dictionary and a list of candidate technical terms (all technical terms in the MREC excluded from the seed set). The double suffix-trie algorithm detects new types in the list of candidates by repeated application of the `type-intro` rule.

### 4.2.1 Creation of Seed Dictionary

The first step in the construction of the seed dictionary involves the application of the C-Value/NC-Value algorithm (Frantzi et al., 1998) on the MREC document collection to extract technical terms (candidate types). I run the implementation of the C-Value/NC-Value method distributed with the Jate framework<sup>6</sup> on the MREC and obtained a list of 2.8 million technical term phrases, which I will refer to as the master list.

---

<sup>4</sup>Named after Armand Borel, the mathematician who introduced the notion.

<sup>5</sup><http://encyclopediaofmath.org/>

<sup>6</sup><https://github.com/ziqizhang/jate>

Each record in the output of the C-Value algorithm corresponds to one technical term – an equivalence class of all variations of the term as observed in the corpus. For example, the following C-Value record

```
flat manifold |Flat manifold |flat manifolds |  
Flat manifolds |Flat Manifolds |  
flat manifold 823.1293931484851
```

is the equivalence class for the technical term “flat manifold”: forms of the technical term “flat manifold” in the MREC are recorded in entries separated by the character “|”. The floating point number at the end of the record is the C-Value termhood score for “flat manifold”.

In the next step, my process selects technical terms that are likely to be types. A fundamental assumption of my method is that technical terms that are types have an entry in at least one online mathematical resource:

1. the Encyclopedia of Mathematics (8730 articles in total at the time of download);
2. Wikipedia articles tagged under the categories “mathematical objects”, “mathematical concepts”, “mathematical structures” and their sub-categories. I used a 2014 Wikipedia dump to access articles under the above categories.

I construct the seed dictionary of types by including those technical terms that entirely match the title of at least one article in the aforementioned encyclopaedias. I consider this a reliable signal that a technical term is a type: if an entire encyclopedia article is devoted to explaining the technical term, then it is likely to be an important mathematical concept. For example, consider the technical term “Riemannian manifold”. The Wikipedia article that describes the concept<sup>7</sup> is titled by the technical term itself, so my method identifies “Riemannian manifold” as a type.

The application of my method to the MREC (Líška et al., 2011) produced a dictionary of 10601 seed types. Table 4.1 shows 18 example types sampled from three segments of the C-Value-sorted seed set (6 types per segment): high, medium and low termhood score.

#### 4.2.2 Gold Standard Evaluation of the Seed Dictionary

My definition of types is intuitive rather than strictly definitional and can therefore be subjective to a certain degree. As a result, I evaluated the quality of accumulated types with the help of 5 human judges (third-year undergraduate and graduate mathematicians). Participants were shown a mixed list of types (as determined by the first algorithm) and non-types (technical terms that had been filtered out by the first algorithm as non-types). Without knowing what the source of each type was, the judges were asked to identify types using a 2-page definition of types (15 rules; listed in appendix C).

The technical term list they were asked to judge consisted of 200 phrases and was constructed by concatenating and randomly shuffling two sub-samples:

---

<sup>7</sup>[https://en.wikipedia.org/wiki/Riemannian\\_manifold](https://en.wikipedia.org/wiki/Riemannian_manifold)

| Type                        | Termhood score |
|-----------------------------|----------------|
| <b>Top-level terms</b>      |                |
| vector field                | 51256.58       |
| vector bundle               | 30062.42       |
| system                      | 25583.11       |
| lie group                   | 25472.74       |
| von neumann algebra         | 14412.15       |
| riemannian manifold         | 13236.66       |
| <b>Med-level terms</b>      |                |
| truncated octahedron        | 28.90          |
| newton polynomial           | 28.26          |
| koenigs function            | 26.76          |
| von karman equation         | 25.57          |
| ball                        | 20.00          |
| enumerable set              | 19.02          |
| <b>Low-level terms</b>      |                |
| conoid                      | -0.23          |
| radially unbounded function | -0.54          |
| klein space                 | -2.45          |
| measuring coalgebra         | -3.75          |
| noetherian space            | -16.77         |
| hermitian function          | -19.27         |

Table 4.1: Examples of identified types grouped by C-Value termhood score.

1. **Sub-sample 1 (Types):** Two-thirds of the sample (134 technical terms) are sourced from the seed list of 10601 phrases. This sample represents the technical terms that the first algorithm classifies as types. Sampling from the seed set allows me to evaluate the decisions made by the first stage of the algorithm, using precision and recall, against human mathematicians. I sampled phrases by distribution of length, in terms of number of words, in order to make the final sample as representative as possible of the technical terms encountered in the mathematical discourse.
2. **Sub-sample 2 (Non-types):** The remainder of the sample (66 phrases) is sampled from the master list of technical terms with two restrictions:
  - (a) sampled phrases are identified as non-types by the first algorithm;
  - (b) sampled phrases come uniformly from three equally-sized segments split according to C-Value score (high, medium and low).

I split the master list into the aforementioned segments and sampled 22 fresh phrases (i.e., phrases not in the seed set) from each by distribution of length. The intention behind this sampling strategy is to present to the judges fresh technical terms that are representative of the diversity of technical terms in the MREC.

The distribution of phrases in the gold standard sample, in terms of phrase length and source,

is shown in Table 4.2. An electronic list of the sample was automatically produced and presented to the judges.

| Source                   | Length of phrase in words |     |    |    |   | Total |
|--------------------------|---------------------------|-----|----|----|---|-------|
|                          | 1                         | 2   | 3  | 4  | 5 |       |
| Sub-sample 1 (types)     | 22                        | 92  | 15 | 2  | 3 | 134   |
| Sub-sample 2 (non-types) | 2                         | 24  | 23 | 13 | 4 | 66    |
| <b>Total</b>             | 24                        | 116 | 38 | 15 | 7 | 200   |

Table 4.2: Distribution of phrase length in the annotation sample.

Precision and recall of the first type detection algorithm is  $P = 73.9\%$  and  $R = 81.8\%$  respectively, resulting in an F-score of 77.7%, with respect to the majority opinion (cf. the confusion matrix in Table 4.3); measured using Fleiss’s Kappa (Fleiss, 1971) the results fall into an intermediate range ( $K = 0.65$ ;  $N = 200$ ,  $k = 2$ ,  $n = 5$ ). The annotators did not receive any training beyond the guidelines and this agreement is neither particularly high nor low. This suggests that the task of identifying types from technical terms, although subjective to a certain degree, is still relatively intuitive to mathematicians. The experiment has also resulted in a gold-standard data set, which can be used by myself or by anybody for evaluating type detection methods.

|               |              | Prediction outcome |          | Total |
|---------------|--------------|--------------------|----------|-------|
|               |              | Type               | Not Type |       |
| Gold Standard | Type         | 99                 | 22       | 121   |
|               | Not Type     | 35                 | 44       | 79    |
|               | <b>Total</b> | 134                | 66       | 200   |

Table 4.3: Confusion matrix for the automatic type detection method.

### 4.2.3 The Double Suffix-Trie Algorithm

My intuition is that the seed dictionary of 10,601 types produced by the first stage of the algorithm is not large enough to cover all the type of all variables in the MREC.<sup>8</sup> In order to overcome this problem, I developed the double Suffix-trie algorithm for expanding the type dictionary, which takes the master list of technical terms and the seed set of types (produced by the first algorithm) as input. The seed set is assumed to contain “known types”, and new types from the master list are confirmed and added by applying the `type-intro` rule. Each step of the double suffix-trie algorithm is illustrated using the example seed dictionary (known types) and the master list (candidate types) shown in Table 4.4.

Initialisation of the double suffix-trie algorithm is done in two steps. First, the phrases in the seed set are tokenised by splitting them at white-space characters. Then, the tokenised known-type phrases are stored on a suffix trie, which I will refer to as the *known types suffix trie*

<sup>8</sup>As we will see in the next chapter, there are 28.6 million sentences in the MREC which contain at least one simple variable and it is highly unlikely that 10,601 types would be enough to cover them.

| Seed dictionary      | Master list                   |
|----------------------|-------------------------------|
| complete coalgebra   | algebra                       |
| riemannian coalgebra | riemannian semialgebra        |
| john steiner         | jo steiner                    |
| simple manifold      | complex and complete manifold |

Table 4.4: Example seed type set (known types) and master list (candidate types).

(KTST). The last word in each type phrase becomes a *terminal node* on the KTST to indicate that it is the end word of a known type. The first word in each type phrase becomes an *end-of-phrase node* to indicate that the path from that node to the root represents a complete known-type phrase.

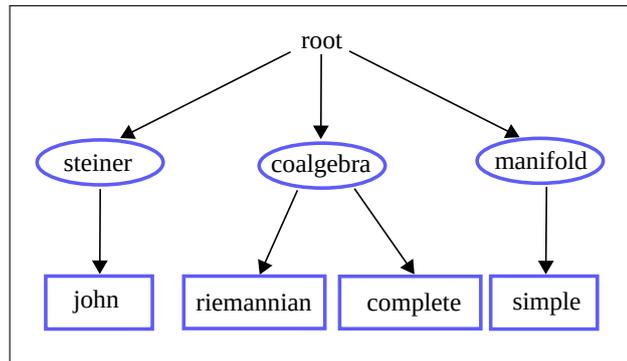


Figure 4.1: Double suffix-trie example: initialisation of the KTST.

The KTST for the seed set in Table 4.4 is shown in Figure 4.1; terminal nodes in the figure are circled and end-of-phrase nodes are enclosed in rectangles. Following initialisation, the double suffix-trie algorithm proceeds as follows.

**Step 1: Construct Contracted Set** Words in terminal nodes on the KTST are assumed to be supertypes (e.g., “coalgebra”, “steiner” and “manifold” in my example, Figure 4.1). I have compiled a dictionary of 40 prefixes, such as “co”, “quasi”, “multi”, “semi”, “di”, “hypo” and “super”, by manually looking at the master list of technical terms. My algorithm uses this list to apply *prefix transformation* to terminal nodes on the KTST as follows. First, all words in terminal nodes are collected from the KTST. Then, all prefixes in the prefix list are removed from the collected words. This produces a new list of candidate supertypes, which I will refer to as the *contracted set*. New words produced by this transformation are added to the KTST as terminal nodes.

**Step 2: Apply Prefix Expansion** Additional candidate types are produced by applying prefix expansion twice on the words in the contracted set. New types are formed by expanding the contracted set in a tree of up to depth 2. At each level of this tree, new types are created by pre-pending the prefixes in the prefix list to all words in the previous level of the expansion tree.

In the context of my example, this step is illustrated in Figure 4.2: the type “coalgebra” is contracted to “algebra”. Although prefix expansion is applied to every word in the contracted set,

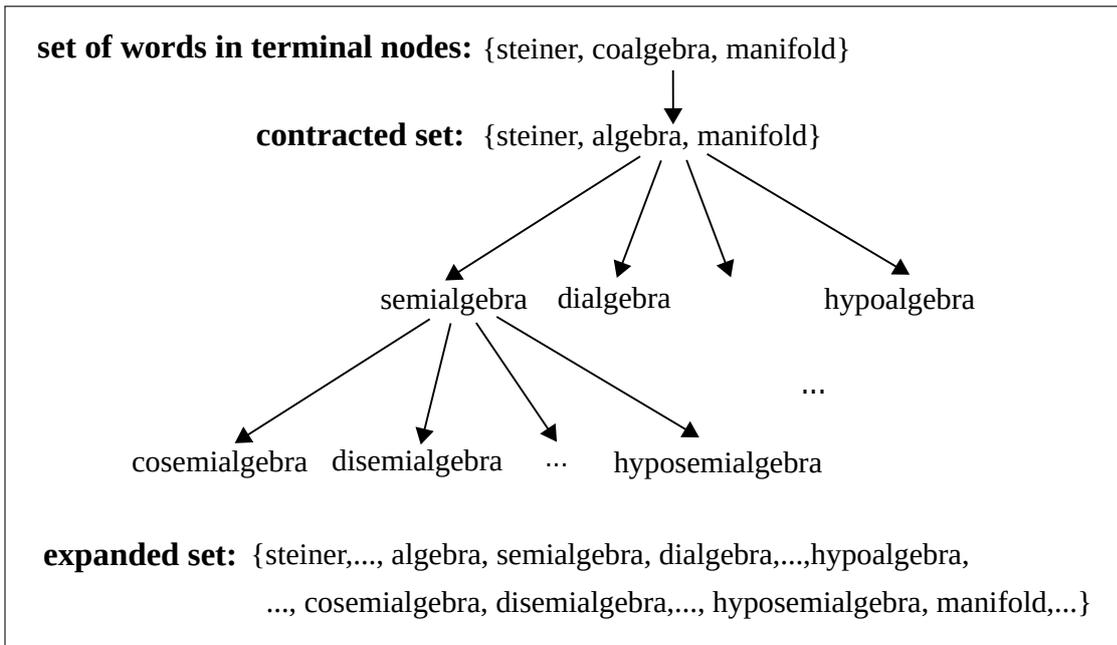


Figure 4.2: Double suffix-trie example: application of prefix transformations and type expansion.

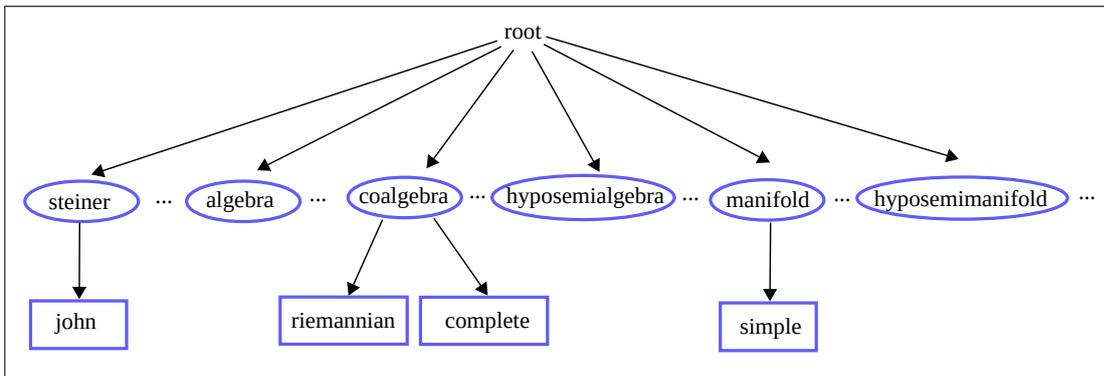


Figure 4.3: Double suffix-trie example: The CTST at the end of step 2.

in Figure 4.2 I only show the process for the word “algebra”: level 1 on the expansion tree for “algebra” contains words generated by pre-pending prefixes to the word itself (level 0). Types at level 2 in the example are generated by pre-pending all prefixes in my list to the words in level 1, as shown in the figure.

**Step 3: Copy Tries** Words of terminal nodes in the KTST and words generated in the previous step (the expanded set in Figure 4.2) are added onto a fresh suffix trie, which I refer to as the *candidate types suffix-trie* (CTST). At this step, all nodes on the CTST are marked as terminal. Then, the KTST is copied on the CTST. For example, at the end of this step the CTST for the example from Table 4.4 might look like the suffix trie in Figure 4.3.

**Step 4: Treat Author Names** My algorithm uses a list of known author names obtained from MREC meta-data to search for author names on the CTST. Phrases on the CTST are paths from end-of-phrase nodes (denoted by the rectangles in the above figures) to terminal nodes (denoted



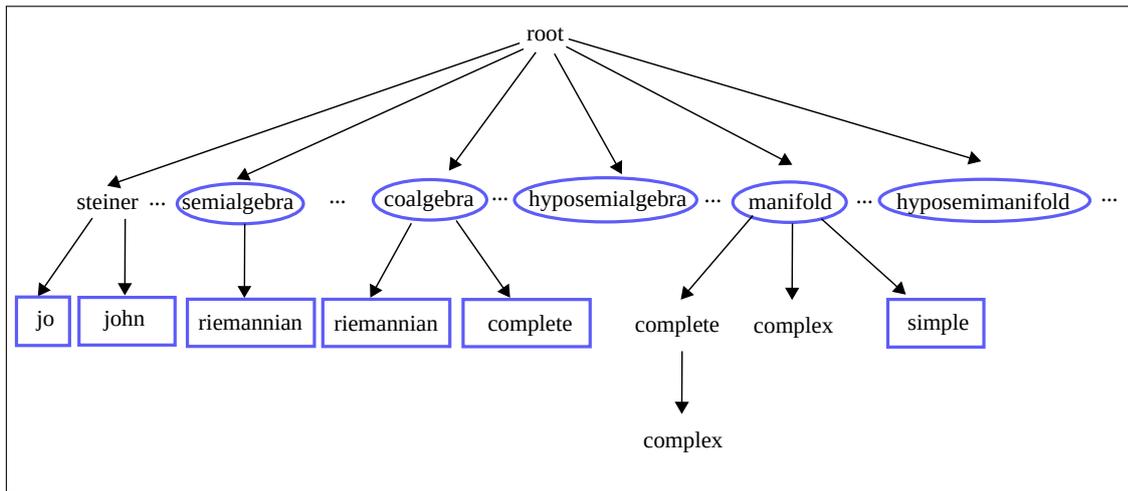


Figure 4.4: Double suffix-trie example: The CTST at the end of step 4.

{ algebra, complete coalgebra, riemannian coalgebra, simple manifold, riemannian semialgebra }

To illustrate why these types specifically are generated by my algorithm, consider the CTST in Figure 4.4. The phrase “jo steiner” is not a type because the parent node of the word “jo” – “steiner” – is a non-terminal one. The candidates “complex manifold”, “complex complete manifold” and “complete manifold” are also identified as non-types by my method. This is the desirable behaviour because the nodes “complex” and “complete” under the “manifold” sub-tree are not marked as end-of-phrase nodes. The reasoning behind this exclusion is as follows: candidate technical terms can be types if and only if they are explicitly listed in the input master list, i.e., the C-Value method detects them as technical terms. Candidates generated at step 4 that are not technical terms detected by the C-Value method cannot be types.

Step 6 is the type dictionary-wide application of the `type-intro` rule. The KTST is the knowledge-base of existing types – known types are represented by terminal nodes. New types are generated from the CTST by successfully discovering a premise for the `type-intro` rule for each technical term type candidate. For example, the candidate phrase “riemannian semialgebra” is introduced to the type dictionary by the following reasoning:

1. “coalgebra” is a known type on the KTST (Figure 4.1).
2. “semialgebra” is a type derived from “coalgebra” in step 2 (Figure 4.3).
3. “riemannian semialgebra” is a candidate in the master list, and is determined to be a type by the application of the `type-intro` rule:

$$\text{type-intro} \frac{\text{istype}(\text{semialgebra}) \wedge \text{riemannian semialgebra} \rightarrow_s \text{semialgebra}}{\text{istype}(\text{riemannian semialgebra})}$$

In step 6, my algorithm determines that

riemannian semialgebra  $\rightarrow_s$  semialgebra

because the candidate “riemannian semialgebra” in the master list has “semialgebra” as its suffix: the node for “riemannian” in Figure 4.4 has the terminal node “semialgebra” as its parent.

I run the double suffix-trie algorithm with the seed set of 10601 known types (produced by the first stage of the type detection algorithm) and the master list of 2.8 million technical terms (obtained by running the C-Value method on the MREC) as input. This produced a list of 1.23 million types.

### 4.3 Type Embeddings

In order for mathematical IR to link queries and documents effectively, it is important to find mathematical concepts that are related (e.g., by type/sub-type relationship), even in cases where this is not expressed on the surface. I use type embeddings for this task. A type embedding space is a word embedding space (section A.1.4) that includes distributional vectors for types.

**input sentence:** "Let  $x$  be a complete Riemannian manifold."

**step 1:** Let @@@ be a complete Riemannian manifold.

**step 2:** <start>, let, @@@, be, a, complete, riemannian, manifold, ., <end>

**step 3:** <start>, let, @@@, be, a, complete, riemannian, manifold, ., <end>

**step 4:** <start>, let, @@@, be, a, \_\_complete\_riemannian\_manifold\_\_, ., <end>

**rewritten sentence :** "let @@@ be a \_\_complete\_riemannian\_manifold\_\_."

Figure 4.5: Example: rewriting sentences in MREC documents to produce type embeddings.

I produce a type embedding space by applying a transformation to the MREC collection I call *rewriting*. Rewriting takes two inputs: (a) the MREC (or any mathematical corpus) and (b) a list of types obtained by my type extraction algorithms. The output of the rewriting transformation is a copy of each document in the MREC with all multi-word types replaced with atomic tokens.

I will now describe the process of rewriting using an example. Suppose that the input document is the sentence

“Let  $x$  be a complete Riemannian manifold”

and that the list of types contains the phrases “manifold”, “riemannian manifold” and “complete riemannian manifold”. I perform rewriting as follows:

**Step 1:** For each document in the MREC, I replace all mathematical expressions (MathML blocks) with a dummy token (“@@@”). For example, the output of this step for the input sentence in my example is as shown in step 1 of Figure 4.5.

There are two reasons why I eliminate MathML during the rewriting process. First, my type embedding space will have one vector representation for all mathematical formulae – a distributional vector that represents the relationship between mathematical expressions in general and their surrounding words. Second, mathematical symbols and MathML tags are not important for the type embedding space and can be omitted to optimise its construction.

**Step 2:** I sentence segment and word-tokenise each MREC document using the Stanford toolkit normalising all words to lower-case. The output at step 2 for the example input is shown in Figure 4.5.

**Step 3:** For each word-tokenised sentence, I identify sequences of words that match types in the type list. I use a finite-state transducer (FST) based on the Aho-Corasick algorithm (Aho and Corasick, 1975) to match all types on the sequence of tokens in a sentence in one pass. I match types in a longest-match-first manner on the token sequence of the sentence.

For example, the red boxes in step 3 of Figure 4.5 show all possible matching types on the input token list: “manifold”, “riemannian manifold” and “complete riemannian manifold”. However, the type I consider to be a successful match with the token sequence is the longest type: “complete riemannian manifold”.

**Step 4:** I merge the word tokens of each successfully matching type into a single token. I place a “\_” character between distinct words of a multi-word type. I also add a “\_” string to the beginning and end of the new token to indicate that it represents a multi-word type.

As shown in Figure 4.5, the token sequence “complete”, “riemannian”, “manifold” is replaced by the single token “\_complete\_riemannian\_manifold\_”.

**Step 5:** I generate a string representation of the rewritten sentence. I concatenate the updated token sequence from step 4 and place a whitespace character between tokens, as shown in Figure 4.5.

**Step 6:** The rewritten sentences for a given MREC document are written into a fresh file and is ready to be processed by any retrieval engine.

I applied the rewriting transformation to the MREC using the type list of 1.23 million types produced by the double suffix-trie algorithm (section 4.2.3). I passed all rewritten documents to `word2vec` (Mikolov et al., 2013a; Mikolov et al., 2013c) in skipgram mode with negative

sampling (SGNS) and window size=10, to compute the type embedding space. A detailed description of the `word2vec` skipgram method is presented in Appendix A.1.4.

My type-based retrieval models interface to the type embedding space using `gensim` (Řehůřek and Sojka, 2010) in Python. I use `gensim`'s `most_similar` method to retrieve the top  $n$  most similar types or words to a particular type or word. For example, Table 4.6 shows the top 10 most similar types (in boldface) and non-type words (non-bold) to the input types “polynomial” and “lie group”.

| “polynomial”                    |            | “lie group”                            |            |
|---------------------------------|------------|--|------------|
| Word/type                       | Similarity | Word/type                              | Similarity |
| <b>rational function</b>        | 0.922      | <b>compact lie group</b>               | 0.913      |
| <b>linear form</b>              | 0.899      | <b>lie algebra</b>                     | 0.91       |
| <b>linear polynomial</b>        | 0.896      | <b>homogeneous space</b>               | 0.899      |
| <b>symmetric polynomial</b>     | 0.885      | <b>symmetric space</b>                 | 0.898      |
| <b>monomial</b>                 | 0.884      | <b>complex semisimple lie group</b>    | 0.875      |
| <b>formal series</b>            | 0.875      | <b>infinite dimensional lie group</b>  | 0.873      |
| <b>rational polynomial</b>      | 0.873      | <b>transformation group</b>            | 0.871      |
| monic                           | 0.873      | <b>nilpotent lie group</b>             | 0.868      |
| <b>reduced polynomial</b>       | 0.873      | <b>riemannian symmetric space</b>      | 0.864      |
| <b>trigonometric polynomial</b> | 0.871      | <b>invariant differential operator</b> | 0.863      |

Table 4.6: Example: Top 10 most similar words/types to the types “polynomial” and “lie group”.

Table 4.6 demonstrates the usefulness of the type embedding space. First, supertype/subtype relationships between types that are not reflected lexically can be captured by close similarity in the type embedding space. For example, in Table 4.6, the type “monomial” is similar to the input type “polynomial” in the type embedding space – a “monomial” is an instance of “polynomial” with only one term.

Second, similarity in the type embedding space can be used to identify related concepts that do not have a common suffix on the KTST. In Table 4.6, for example, the types “lie algebra” and “homogeneous space” are related to the input type “lie group” despite the fact that the types do not share a common suffix:

- Every Lie group has an associated Lie algebra<sup>9</sup>;
- A Homogeneous space is a space with a transitive group action by a Lie group<sup>10</sup>.

#### 4.4 Retrieval based on Textual Types

The first hypothesis in this thesis is that research level math IR will benefit from treating mathematical types in text as atomic units. In the context of the test collection paradigm (Section 2.3.1, Chapter 3), the probability ranking principle (Section 2.3.2) and text retrieval in general (Section 2.1), this hypothesis can be broken down into the following questions:

<sup>9</sup>[https://en.wikipedia.org/wiki/Lie\\_group#The\\_Lie\\_algebra\\_associated\\_with\\_a\\_Lie\\_group](https://en.wikipedia.org/wiki/Lie_group#The_Lie_algebra_associated_with_a_Lie_group)

<sup>10</sup><http://mathworld.wolfram.com/HomogeneousSpace.html>

- (1) Is overlap of mathematical concepts, rather than individual words, a better indicator of topical similarity between queries and documents in research-level mathematics? This question is related to query expansion. I hypothesise that concepts in research-level mathematical queries relate strongly to those in relevant documents.
- (2) Under the bag-of-terms IR paradigm, are labels of mathematical concepts better signals of topical similarity between queries and documents than individual words? My hypothesis is that references to mathematical concepts, when regarded as atomic tokens, are more important than words in text-based retrieval of research-level mathematical information needs.

So far in this chapter I have introduced four resources and techniques that are necessary to investigate this hypothesis:

- I a type dictionary of 1.23 million types, produced automatically, that lists all mathematical concepts in the MREC;
- II a model of subtype and supertype relationships between types using a suffix trie;
- III a model of relatedness between types in a type embedding space;
- IV a rewriting technique (section 4.3) that replaces multi-word types with atomic tokens in a given textual resource or large corpus, such as the MREC.

In section 2.1 we have seen that there are two paradigms in textual retrieval: heuristic-based retrieval and retrieval based on language modelling. Standard models, such as VSM, BM25 and multinomial language models, are unigram models based on the bag-of-words IR paradigm – retrieval is performed using statistics about individual terms. In order to investigate my first hypothesis using standard, text-based IR models I adopt three strategies:

1. I re-purpose the rewriting technique to: (a) rewrite CUMTC queries such that type phrases are replaced by atomic type tokens in the dictionary of types derived from the MREC and (b) to construct type-aware Lucene indexes of the MREC.
2. I up-weight the significance of types by applying *2X boosting*: the value stored in the term vector of each document for a type token (i.e., the type's term frequency for that document) is doubled at index time.
3. I use the type embedding space for type-based query expansion. I add types to CUMTC queries that are related, according to the type embedding space, to the types already present in the queries.

The first strategy addresses sub-question (1) directly. Specifically, it allows me to model similarity between queries and documents based on types using standard unigram retrieval retrieval models: types become terms, rather than multi-word phrases. Furthermore, by applying the rewriting technique on queries and during document indexing, I normalise mathematical concepts in my queries with types observed in the MREC. This allows me to model concept overlap between queries and documents consistently.

The second strategy is my method of elevating the significance of types over general document terms in order to investigate sub-question (2). Term frequency is monotonically related to the score assigned to a document given a particular query in retrieval models from both paradigms.

The third strategy allows me to compare the effectiveness of types in text-based retrieval with state-of-the-art query expansion techniques, such as PRM2 (Lv and Zhai, 2011) and RM3 (Abdul-jaleel et al., 2004; Lv and Zhai, 2009).

I implement strategies 1 and 2 in a unified pipeline, shown in Figure 4.6. I use the pipeline to discover all occurrences of dictionary types in MREC documents and CUMTC queries and to index types and terms in MREC documents.

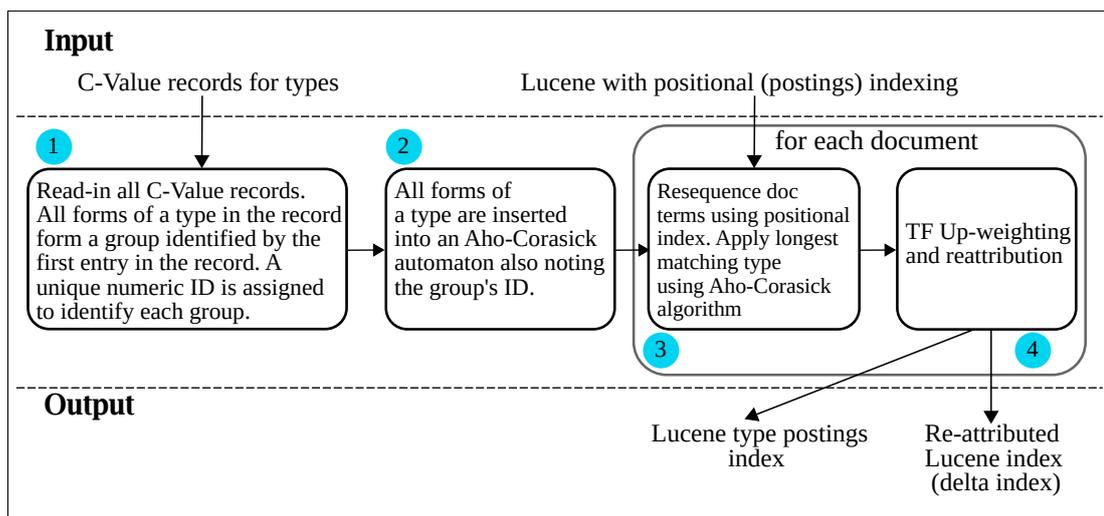


Figure 4.6: The typed indexing pipeline implementing strategies 1 and 2 on the MREC.

The pipeline takes two data sets as main inputs and the target locations for the two output indexes as secondary inputs. The main inputs to the pipeline, as shown in the top row of Figure 4.6, are: (a) The list of types as a sequence of C-Value records and (b) a Lucene index of the MREC produced with positional indexing enabled.

The pipeline assumes that the input positional index has at least two fields. The first is the “contents” field which stores the original content of each indexed document. This field must also have a positional index attached to it along with a term vector that can be accessed through the Lucene API. A field labelled “resourceName” is expected to keep the ArXiv ID of the indexed document. In order to process the input text, I have produced custom Lucene analyzers that preserve dashes and underscores, tokens that are frequently part of technical terminology.

| Field      | Description   | Example             |
|------------|---|---------------------|
| term       | The form of a type as it appears in the document.                           | __spanning_vector__ |
| termleader | The C-Value record group key for the term                                   | spanning vector.    |
| arxivid    | The ArXiv ID of the source document.  | 0704.0159           |
| docid      | The document's numeric ID in the input Lucene index.                        | 187871              |
| posting    | The positions in the source document the index form of the type appears at. | 561,970,1210        |
| tf         | The term frequency of the form in the document.                             | 3                   |

Table 4.7: Fields in the type postings index.

|   |
|---|
| <b>Original text</b><br>a Riemannian manifold is a smooth manifold  |
| <b>Original term vector</b><br>(a,2), (Riemannian,1),(manifold,2),(is,1),(smooth,1)   |
| <b>Types in the input stream</b><br>Riemannian manifold, smooth manifold  |
| <b>Re-Attributed term vector</b><br>(a,2), (__Riemannian_manifold__,1),(is,1)(__smooth_manifold__,1)                              |
| <b>Re-generated delta index text</b><br>a a __riemannian_manifold__ __riemannian_manifold__<br>is smooth_manifold smooth_manifold |

Table 4.8: Example of re-attribution and delta index.

The output type postings index contains an entry for each form (i.e., entry in the type's C-Value record) a particular type appears in every document. In addition, the position where each instance occurs (postings) is also recorded. The output "delta index" maintains these two fields, but the data in the "contents" field is modified with type occurrences doubles (2X boosting).

The first two steps of my pipeline initialise the data structures necessary to apply re-writing for typed textual indexing. Re-writing for indexing keeps track of where each form of a type appears in the source document for the purpose of building a type postings index. In step 3, an MREC document  $d$  is re-written and a term vector containing the types and words of the document is produced. Re-attribution is performed in step 4 as follows. First, I decrease the term frequency of each word captured by an instance of a type. Then, I generate a token for each type as described in Section 4.3, which uniquely identifies each instance of a particular type. If the generated word exists in the term vector, I increase its term frequency by 1. If the new word is not in the term vector, I add it and set its term frequency to 1.

Once re-attribution for the document is completed, I update the output Lucene indices. For every type form in a document, I generate an entry in the Lucene type postings index that contains the fields shown in Table 4.7.

I generate one entry for the document in the "delta index" by (a) modifying the term frequency of types by an up-weighting factor (2X) and (b) re-generating the text of the "contents" field to reflect the changed frequencies.

Table 4.8 shows an example of the application of step 4. The first row in Table 4.8 is the input document and the second row is its term vector. In step 3, two types are detected: “Riemannian manifold” and “smooth manifold”. Step 4 generates the re-attributed term vector shown in the fourth row of Table 4.8. The last row in Table 4.8 shows the text generated after up-weighting types by a factor of 2 that is stored in the delta index.

The “delta index” implements strategies 1 and 2 and allows me to carry out type-based textual retrieval experiments with models based on the bag-of-words paradigm efficiently using Lucene. I implemented my pipeline in Java in a data-parallel manner: many documents can be processed (steps 3 and 4) at the same time in different execution threads.

Having discussed the ideas behind types and the resources required to identify them in text, I now introduce two textual retrieval models based on types, which will be part of my experimental investigation in Chapter 7.

**Types2X** Lucene VSM with 2X type boosting. I apply longest matching to a Lucene positional index and emit a type-aware “delta index”. This type discovery and normalisation pre-processing step is also applied to queries. My assumption is that types are a valuable source of information for an MIR system. Types2X assumes the role of a type-aware model that performs the simplest manipulation of those types that are physically present in the query (simple 2X boosting). Therefore, in my comparison, Types2X is used to measure the simplest possible way of incorporating types during retrieval (as opposed to just using terms).

**Types2XExp** Lucene VSM with 2X type boosting and type-based query expansion. Unlike mathematical papers, queries are not always rich in types: on average each query contains around 13 type instances while documents in the MREC contain on average close to 548 type instances. Types2XExp overcomes this problem by enriching queries with types. Queries are expanded using the types (as opposed to the terms) they contain.

For each type in a query, the type-embedding space (using word2vec similarity as discussed in section 4.3) is used to discover  $n$  fresh related types. Semantic relatedness between types is modelled by the cosine similarity of their vector representations.

The set of fresh types is appended to the original query and the new query is executed on a 2X type up-weighted VSM. The value for  $n$  is the only parameter of the model, which has been experimentally set to  $n = 5$ .

In my experimental evaluation of type-based textual MIR models (Section 7.1), I compare Types2XExp to state-of-the-art textual retrieval models with query expansion based on the bag-of-words paradigm. In doing so, my goal is to ascertain that any observable improvements in retrieval efficiency can be attributed to type-based query expansion specifically, rather than query expansion in general.

## 4.5 Comparison with Related Work

I use types to model mathematical concepts, but other constructs have been proposed in the literature for the same purpose. The task of extracting types from the mathematical discourse is closely related to extracting “denotations” (labels connected to term clouds) for symbols (Grigore et al., 2009; Wolska and Grigore, 2010; Wolska et al., 2011) and “descriptions” (coreferences between formulae and text) for mathematical formulae (Quoc et al., 2010; Kristianto et al., 2012; Kristianto et al., 2014b). The main difference to my work is that denotations for specific formulae come from the context surrounding formulae, whereas types are extracted independently of formulae.

The idea behind denotations is to model the semantics of formulae using manually and semi-automatically constructed *term clusters* (Grigore et al., 2009). A term cluster for a concept is composed of a label and a bag of nouns. Grigore et al. (2009) take operators listed in OpenMath content dictionaries (CDs) to be mathematical concepts and extract nouns from operator dictionary descriptions to model their semantics. This set of nouns is enriched manually using additional terms taken from the University of Cambridge mathematical thesaurus and the MathWorld lexicon of mathematical terms. Table 4.9 shows the term cluster for “algebraic structure” (adapted from Grigore et al. (2009)).

| Term Cluster Name   | Term Cloud   |
|---------------------|--|
| algebraic structure | algebra, array, basis, field, generator, group, groupoid, ideal, lattice, matroid, monoid, quaternion, ring, semigroup, space, subfield, submonoid, subsemigroup,... |

Table 4.9: The “algebraic structure” term cluster, adapted from Grigore et al. (2009).

Quoc et al. (2010) extract descriptions for formulae (phrases or sentences) by matching one or more of the patterns shown in Table 4.10 to the surrounding context of formulae. The patterns in Table 4.10 are matched as templates on sentence parse trees, where CONC, DESC and FORM are the portions of the context expected to correspond to concepts, description and formulae respectively.

| Rule | Pattern                          |
|------|----------------------------------|
| 1    | CONC is DESC: FORM               |
| 2    | CONC IS DESC. In our case FORM   |
| 3    | CONC IS DESC. So, ..., FORM      |
| 4    | CONC FORM                        |
| 5    | CONC is denoted by FORM          |
| 6    | CONC is given by ... FORM        |
| 7    | CONC can be written as ...: FORM |
| 8    | FORM where CONC is DESC          |
| 9    | FORM satisfies CONC              |

Table 4.10: The patterns used by Quoc et al. (2010) to extracting descriptions for formulae.

Kristianto et al. (2012,2014b) refined the patterns by Quoc et al. (2010) into a set of seven templates, defined as regular expressions, that are matched on sentence parse trees.

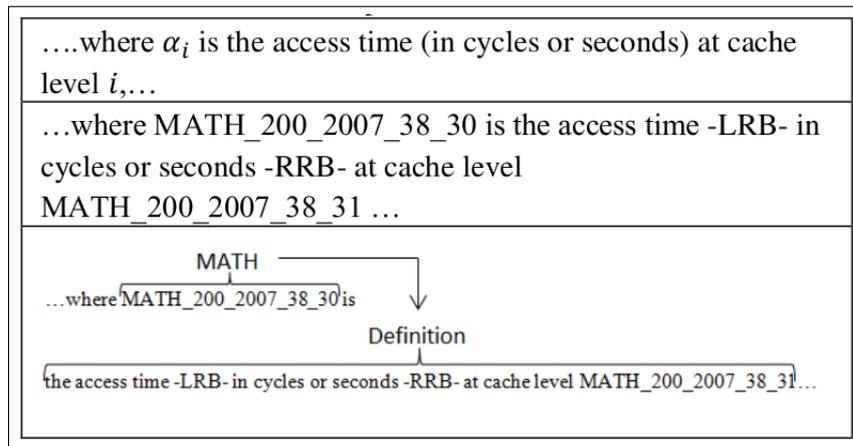


Figure 4.7: Example of extracting descriptions adapted from Kristianto et al., 2012.

Figure 4.7 shows an example of formula description extraction. The first row of Figure 4.7 is the source sentence, where the variable  $\alpha_i$  corresponds to cache level  $i$  access time. The second row is the output of the pre-processing step whereby symbols are substituted with unique identifiers (e.g.,  $a_i$  becomes MATH\_200\_2007\_38\_30). The last row of Figure 4.7 shows how the source sentence is segmented into the symbol reference (MATH\_200\_2007\_38\_30) and the symbol’s definition (“the access time -LRB- in cycles or seconds -RRB- at cache level MATH\_200\_2007\_38\_31”).

The authors constructed an SVM classifier (Section A.2.1) for description extraction and compared their refined rules and SVM against a “nearest noun” baseline.

Types are related to term clusters and descriptions in the sense that they can be used to assign meaning to formulae from the text. However, types incorporate the advantages of both constructs. Like term clusters (Grigore et al., 2009), types are labels for mathematical concepts that have a distributional interpretation, for instance via embedding vectors as discussed in section 4.3. Unlike term clusters, types can be constructed automatically, without manual curation. Types share this advantage with descriptions, which are also extracted automatically (Quoc et al., 2010; Kristianto et al., 2012; Kristianto et al., 2014b).

However, my approach to modelling mathematical concepts using types has additional advantages over term clusters and descriptions:

- Term clusters model relatedness between mathematical concepts up to two levels of abstraction. The first level is the explicit relatedness between terms in the term cloud of a cluster. The second level is the implicit relatedness between the terms in the cloud and the abstract mathematical concept represented by the cluster itself.

Types, in contrast, model subtype/supertype relationships and topical relatedness of arbitrary complexity through (a) a suffix trie and (b) a type embedding space (section 4.3).

- Quoc et al. (2010) and Kristianto et al. (2012,2014b) model mathematical concepts strictly in relation to formulae in the discourse. For example, Kristianto et al. extract descriptions with respect to the local context of a specific formula. In contrast, I use types to model mathematical concepts globally in a manner that is decoupled from the formulae in the discourse – types exhibit distributional characteristics independently of formulae. I use the distributional profile of types to investigate my first hypothesis.
- Although in some cases descriptions extracted by Kristianto et al. (2012, 2014) are noun phrases and resemble types, they often incorporate details that are particular to a specific context. For example, the extracted description for formula  $a_i$  presented in Figure 4.7 contains information that is specific to the context (i.e., time at a particular cache level measured in cycles or seconds).

In contrast types, being technical terms, have predictable structure. As a result, types are labels that globally and uniquely identify mathematical concepts in the discourse in a context-independent manner. This allows me to construct typed retrieval models that match formulae in a flexible manner through informal inference, such as type unification (Chapter 6).

## 4.6 Chapter Summary

In this chapter I introduced mathematical types: special phrases in the mathematical technical terminology that label mathematical structures, objects and notions instantiated in the mathematical discourse as variables. As technical terms, types have predictable structure and are often compositional – longer types are specialisations of shorter types contained in their suffix.

In the context of this work, this observation has two implications. First, it allows me to make inferences about the relatedness between types using a suffix trie. Second, identifying types in queries and documents allows me to build a type embedding space and to discover topically related types that do not have a common suffix.

In section 4.2, I presented an algorithm for automatically constructing a type dictionary from a corpus of mathematical documents. As part of this work, I produced three resources: a gold-standard data set for evaluating type detection methods, a dictionary of 1.23 million types and a type embedding space (Section 4.3).

There are three differences between my approach and prior work to modelling mathematical concepts:

1. Unlike term clusters, types model the subtype/supertype relationship and topical relatedness of mathematical concepts at an arbitrary level of complexity.
2. Types model mathematical concepts without attachment to the formulae in the discourse. Unlike descriptions (Quoc et al., 2010), which are explicitly bound to an attached formula, types can globally model the distributional characteristics of mathematical concepts.

3. Types are conducive to shallow type inference about variables. For one, the compositional structure of types simplifies identifications of subtype/supertype relationships between types. Second, the distributional characteristics of types as observed in a large corpus, can be used to model topical relatedness between types.

In my type-based textual retrieval experiments, I use a type embedding space (Section 4.3) to expand queries with related types. I used both approaches to define shallow type inference rules that are the foundation of my typed retrieval models (Chapter 6). In the next chapter I will introduce variable typing – a task that is a prerequisite to typed retrieval.



# Chapter 5

## Variable Typing

If types are semantic labels that act as denotations, then variable typing can be defined as the task of assigning types as denotations to variables. Typing is a prerequisite for my typed retrieval models (Chapter 6), which require types for all formulae. I use supervised ML to perform variable typing as a binary classification task, after creating a test and training set for the task. The best-performing classifier found in the current chapter will be applied to my test collection CUMTC in order to type variables.

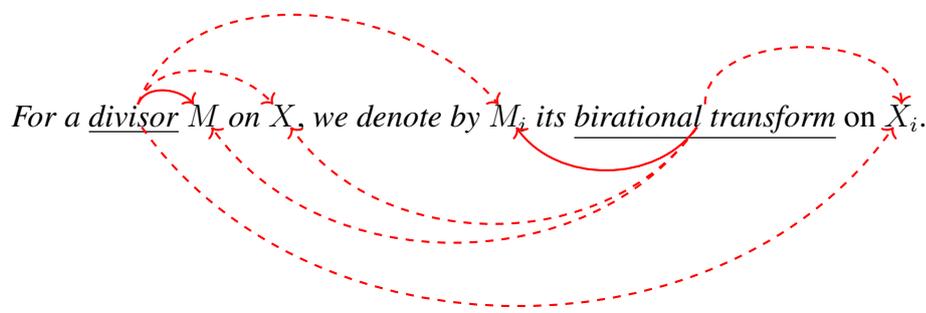
The variable typing data set and results have been published in the proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT 2018), (Stathopoulos et al., 2018)<sup>1</sup>.

There is an interaction between the symbolic and textual contexts of types, whereby meaning is assigned to variables by the surrounding text in the same sentence. To illustrate how this works, let us assume that a sentence contains a pre-identified set of variables  $V$  and type instances  $T$ , and all edges  $(v, t) \in V \times T$  between them. The goal is to predict which of these edges constitute valid *typings*. An edge is a typing if type  $t$  is a denotation (i.e., assigns meaning) to variable  $v$  in the sentence. (Note that this independent definition does not model the uniqueness constraint that each variable can at most have one type.) If  $(v, t)$  is a typing, then  $v :: t$ <sup>2</sup>. For example, the sentence

---

<sup>1</sup>Statement of Collaboration: my co-authors on this publication did the following work: With respect to data construction, Simon Baker co-designed the annotation scheme and annotated 550 sentences. He also built and trained the convolutional network used in the paper. Marek Rei built and trained the biLSTM used in the paper, and also provided typing predictions over the entire MREC. All other work, including the reimplementing of all baselines, formatting of the input for the neural networks and the overall design of the experiment, was performed by me.

<sup>2</sup>I follow notation from chapter 4, where  $v :: t$  means that  $v$  has type  $t$



has eight candidate typings in  $T \times V$ , where  $T = \{\text{divisor, birational transform}\}$  and  $V = \{M, X, M_i, X_i\}$  as indicated by the arrows. In the annotated sentence above, arrows drawn with solid lines represent typings (positive edges). So, the variables  $M$  and  $M_i$  have types “divisor” and “birational transform” respectively. In contrast, arrows drawn with dashed lines represent edges that are not typings (negative edges). The edges ( $M$ , “birational transform”) and ( $X$ , “divisor”), for example, are not typings.

In this chapter, my definition of “variable” mirrors that of “simple variable” proposed by Grigore et al. (2009): formulae in the discourse are considered to be “variables” if they are composed of a single identifier with scripted expressions of arbitrary complexity. For example, the formulae

$$M_{i+1}, X \text{ and } e^i$$

are variables. In contrast, the formula

$$f(x) = e^x$$

is not a variable because it is composed of the sub-expressions  $f(x)$  and  $e^x$ , which are connected by the binary operator “=”. Later in this chapter (Section 5.4), I will extend this definition of variables to accommodate more complex mathematical expressions.

The meaning of mathematical text is conveyed through the interaction of two modalities: the textual modality (flowing text) and the mathematical (or symbolic) modality (mathematical formulae) (Ganesalingam, 2008). Variable typing models one particular interaction between the two modalities: the assignment of meaning to the symbolic modality by the textual. My variable typing task makes five simplifying assumptions about this interaction:

1. Typings occur at the sentential level – variables in a sentence are assigned types from that sentence as their denotation;
2. each variable is assigned one type in the sentence as its denotation but types can become denotations to multiple variables within the scope of the sentence;
3. variables and types in the sentence are known a priori;
4. the decision on whether one edge is a typing or not is made independently of other edges in the sentence;

5. typing decisions about edges in one sentence are made independently of decisions made in other sentences – given a variable  $v$  in sentence  $s$ , type assignment for  $v$  is agnostic of other typings involving  $v$  from other sentences.

The motivation behind the assumptions above is as follows.

- Assumption 1 is motivated by empirical studies. Wolska and Grigore (2010) have shown that 67.4% of symbols in mathematical documents are explicitly declared within the first five mentions. The majority of those, 58% according to Wolska and Grigore, are declared in the sentence they are first introduced. Gödert (2012) confirms the locality of variable declarations to introductions.
- Assumptions 1, 4 and 5 simplify the task of assigning denotations to symbols by constraining the search to the local, sentential context. However, these assumptions also introduce a fine granularity to the task: it is possible to make global (document-level) decisions on the denotations of symbols using local (sentence-level) decisions.
- Assumption 2 allows for references to the same variable to be assigned different types in different sentences. In my design, the task of disambiguating multiple type assignments coming from different sentences is a separate task performed downstream. I delegate this task to a type disambiguation algorithm (Section 6.3.1) that aggregates local typing decisions and disambiguates type assignment at the document level.
- Assumptions 1, 4 and 5 are also motivated by the fact that the mathematical discourse is composed of numerous local contexts, such as theorems, lemmas and proofs (Ganesalingam, 2008).
- Assumption 3 is a technical requirement for developing supervised machine learning models and for applying them at scale to the task.

Variable typing is performed as a prerequisite task to typed model building: by associating types in the textual modality to variables, I can create SLTs with typed constituents for each document and query in the CUMTC (Section 6.3.2).

Formulae with typed constituents enable typed retrieval models to apply the rules for type inference and unification (Chapter 4 and Section 6.2, respectively) while estimating similarity between the symbolic contexts of queries and documents.

In the next section, I discuss my gold standard variable typing data set (collaboratively constructed with Simon Baker). In section 5.3, I introduce two baseline models for variable typing and propose three new machine learning models for automatic typing. Evaluation of my models and a discussion of their performance against the baselines is presented in section 5.3.2.

## 5.1 A Variable Typing Data Set

The design for sampling, labelling and annotating the variable typing data set is influenced by three characteristics of the task and data.

First, assumptions 1, 4 and 5 of the variable typing task imply that the smallest unit of sampling and annotation is a sentence. Therefore, sentences can be sampled independently and annotators can look at individual sentences in any order, without the need to annotate entire documents.

Second, for the data set to be useful for training machine learning classifiers, annotators must annotate every variable–type pair in each sentence – a requirement that follows from the definition of the task.

Third, in the context of variable typing, there is one class of variable–type pairs that is not useful: instances of *type parametrisation*. Type parametrisation is a syntactic phenomenon: a small symbolic expression is embedded in the string name of a type. For example, the correct set of candidate edges for the sentence

We now consider the  $\mathbf{q}$ -exterior algebras of  $\mathbf{V}$  and  $\mathbf{V}^*$ , cf. [21].



should include  $(\mathbf{V}, \text{exterior algebra})$  and  $(\mathbf{V}^*, \text{exterior algebra})$  and not  $(\mathbf{q}, \text{exterior algebra})$ . However, this can be hard to detect automatically because in the data they are represented as a concatenation of strings and MathML. Despite me doing the best I could using pattern matching based on regular expressions, both wrong and missed candidates can occur in the process. Type parameterisation is a phenomenon that I will come back to when discussing my labelling scheme for variable typing in section 5.1.2.

### 5.1.1 Sentence Sampling

The role, complexity and structure of sentences might vary based on their location (section and/or mathematical block) in the discourse, which implies that the location of sentences might also have an impact on the variable typing task. For example, the following sentence from the introduction of Cho and McCullough (2006) is intended to introduce the main contribution of the paper:

In this work we present a new descriptive theory for the tunnels of tunnel number 1 knots in  $S^3$ .

The sentence from the introduction is short, descriptive, mathematically informal and contains few mathematical types and variables. In contrast, sentences in theorem blocks are formal, long and are composed of mathematical statements, possibly involving many types and variables:

**Theorem 5.3.** Let  $\tau$  and  $\sigma$  be *primitive disks* in  $H$  which intersect transversely. Let  $\tau_1$  and  $\tau_2$  be the *disks* that result from surgering  $\tau$  along an *intersection arc* which is outermost on  $\sigma$ . Then  $\tau_1$  and  $\tau_2$  are primitive.<sup>3</sup>

The effects of sentence structure and complexity to the variable typing task cannot be determined a-priori to sampling for the task. Therefore, I adopted a sampling strategy that controls for the diversity of sentences in the mathematical documents.

The first step in sampling sentences for the variable typing task is to further segment each MREC document into its constituent sections and mathematical blocks. For this, I used the sentence-segmented and type-rewritten version of the MREC I produced for type embeddings (section 4.3).

The XHTML+MathML mark-up of MREC documents encodes the presentation of documents, rather than their section structure. I segmented MREC documents into their constituent sections and mathematical blocks by using pattern matching (e.g., theorems and definitions usually start with bold-face “Theorem” and “Definition”) and exploiting the fact that markup for section/sub-section heading mark-up is consistent in the MREC (e.g.,  $\langle h1 \rangle$  for top-level sections,  $\langle h2 \rangle$  for sub-sections, etc.).

I identified the discourse regions shown in Table 5.1 and modelled the structure of each MREC document as a section tree. Nodes on document trees represent one of the discourse segments shown in Table 5.1 and group the sentences in that particular region of a document together.

| Discourse segment   | Description   |
|---------------------|---|
| Abstract            | Text that is part of the abstract of an MREC document.  |
| Section             | A top-level (level 1) section of an MREC document that may have one or more sub-sections as its children.             |
| Subsection          | A sub-section (level 2) in an MREC document that has a level 1 section as its parent.                                 |
| Subsubsection       | Level 3 section that has a subsection as its parent.  |
| Subsubsubsection    | Level 4 section that has a subsubsection as its parent.   |
| Mathematical blocks |   |
| Definition          | A block of text that is part of a formal mathematical definition.   |
| Remark              | A block of text that forms author comments on a particular result.  |
| Proof               | A block of text that is part of a formal proof.   |
| Theorem             | A block of text that is a statement of a theorem.   |
| Lemma               | A block of text that is a statement of a lemma.   |
| Proposition         | A block of text that states a mathematical proposition proof pending.   |
| Conjecture          | A block of text that states a conjecture. Often, the proof of conjectures is omitted or given later in the discourse. |
| Corollary           | A block of text stating a result that follows from one or more theorems previously proven or stated in the discourse. |

Table 5.1: Discourse segments.

<sup>3</sup>From section 3 (the disk of an irreducible 3-manifold) of Cho and McCullough (2006).

Next, I used the section trees to produce a database of MREC sentences. I identified 28.6 million sentences containing at least one simple variable. I recorded the tokenised text of these sentences, along with the following features, in a Lucene index:

1. *Source document*. The ArXiv ID of the document to which the sentence belongs.
2. *Sentence location*. The location of the sentence in the source document. This field takes one of thirteen labels reflecting the discourse regions shown in Table 5.1.
3. *Depth*. The level at which the sentence appears in the section tree. A sentence in a subsection, for example would have depth 3.
4. *Number of types*. The number of types in a sentence.
5. *Number of Variables*. The number of simple variables in the sentence.
6. *Number of words (length)*. The length of the sentence in terms of total number of words.

Features 2 to 6 characterise sentences in a five-dimensional space. I grouped sentences in the Lucene index with the same values for these features, forming  $N$  bins, and counted their occurrence in the MREC. This enabled me to quantify the distributional characteristics of sentences in my corpus for the variable typing task.

To generate a sample of  $S$  sentences, I randomly sample sentences from the  $N$  bins by distribution. For example, suppose that bin 1 represents sentences sourced from theorem blocks, appear at depth 1, have 2 types and 2 variables and length 7. If bin 1 accounts for 60% of MREC sentences, then 60% of the sentences in the generated sample will be randomly sampled from bin 1.

This sampling strategy accounts for the diversity of sentences in mathematical documents, including sentence variations within each section. The training, development and test samples have been produced via repeated application of this sample-by-distribution strategy over the set of all sentences with variables.

### 5.1.2 Annotation Scheme

The annotation scheme for the variable typing task was developed together with Simon Baker prior to bulk annotation, starting from the five assumptions of the variable typing task, and using  $S = 100$  sentences sampled using the strategy discussed in Section 5.1.1, which we separately annotated. We subsequently discussed the relationships between variables and types identified by us in the sample until we reached a consensus, resulting in the annotation scheme in Table 5.2. The 100 sentences were then removed from the sampling pool and excluded from sampling for the human agreement experiment in (Section 5.1.3) and the bulk annotation (Section 5.1.4).

The first label class is used to characterise positive typings: the edge is labelled with the type instance that is the denotation of the variable in the sentence. The label class contains a variable number of labels per sentence, as many as there are type instances in the sentence.

| Label   | Description  |
|---|--|
| $\langle \text{TYPE}_0 \rangle \dots \langle \text{TYPE}_n \rangle$ | Each of $n$ type instances appearing in the sentence.  |
| Type Unknown  | The type of the variable is not in the scope of the sentence.  |
| Type Present but Undetected   | The type of the variable is in the scope of the sentence but is not in the dictionary.   |
| Parameterisation  | Variable is part of an instance of parameterisation.   |
| Index   | Variable is an instance of indexing (numeric or non-numeric).  |
| Number  | Variable is implied to be a number by the textual context (e.g., “the $n$ -th element...”).  |
| Formula is not a variable   | Label used to mark data errors. For example, in some instances end-of-proof symbols are encoded as identifiers in the corpus and are mistaken for variables. |

Table 5.2: Annotation Scheme for Variable Typing.

The other six labels are fixed across sentences and are used to annotate special typing situations. For example, the label “Type Unknown” is used to indicate that a variable is not assigned a type within the scope of the sentence being annotated. The label “Parameterisation” is used to label variables as misidentified instances of type parameterisation, as some such instances inevitably escaped my automated filters.

### 5.1.3 Human Agreement Experiment

I carried out a human agreement experiment between myself and Simon Baker to investigate the effectiveness of the finalised annotation scheme. For this purpose, I sampled a further 108 sentences with a total 182 edges to measure inter-annotator agreement.

I report annotator agreement for three separate cases due to the nature of the scheme, where a variable number of types appear per sentence. This means that the number and identity of labels to choose from is not fixed across sentences, thus excluding the use of categorical agreement metrics such as Kappa.

The first case reflects whether the annotators agree that a variable can be typed or not by its context. A variable falls into the first category if it is assigned a type from the sentential context and in the latter category if it is assigned one of the six fixed labels from Table 5.2. This can be reported using Kappa.

The second case is for instances where both annotators believe a variable can be typed by its sentential context – the variable is assigned a type by both annotators. In this case, Cohen’s Kappa is not applicable because the number of labels varies: there are as many labels as there are types in the sentence. Instead, I report accuracy as the proportion of binary decisions over each possible pairing (typing) where annotators agree, over all decisions.

Annotation is performed on a web-based, multi-user and multi-sample annotation tool I developed. Its goal was to make manual annotation easier and to enforce variable typing assumptions during annotation. My tool enforces all variable typing assumptions except assumption 4 – annotators are explicitly responsible for enforcing this assumption.



Figure 5.1: Snapshot of the sentence annotation tool.

The tool, shown in Figure 5.1, presents annotators with one sentence at a time. Types are clearly marked using bold font. For each variable in the presented sentence a drop-box is attached for the selection of the appropriate label.

The results of the human experiment were as follows:

Agreement on the decision whether a variable can be typed or not is substantial with Cohen’s  $K=0.80$  ( $N=182, k=2, n=2$ ). Agreement on which of the labels (type) applies is 90.9% (44 edges assigned type by both annotators, 40 of these in agreement).

In the last case, I also consider the subset where both annotators agree that a variable is not a type (i.e., is assigned one of the six fixed labels). Here, agreement was found to be moderate: Fleiss’  $K=0.61$  ( $N=123, k=2, n=6$ ). Agreement in this case is lower than expected. However, this only reflects agreement as to why an edge is negative, not whether an edge is negative or not. Therefore, the lower than expected agreement in this case does not necessarily affect the ability of machine learning models using our annotated data to distinguish between positive and negative edges.

#### 5.1.4 Bulk Corpus Annotation

I generated three partitions of the sentences – one test partition, one for training and one for development. Each partition is sampled by distribution (as described in Section 5.1.1) in order to model training and predicting typings over complete discourse units, such as documents. The training, test and development partitions form a 70%-20%-10% split of all sampled sentences. I carried out the bulk of the annotation myself with the exception of 500 sentences that were annotated by Simon Baker.

The resulting data set contains 7,803 annotated sentences, sampled independently from the MREC, covering 33,524 edges in total. Table 5.3 shows the training/development/test partitioning scheme used.

## 5.2 Comparison with Related Work

Variable typing is the machine learning task of assigning denotations to variables in mathematical text. Denotations in variable typing take the form of mathematical types from a fixed dictionary.

|                       | Train  | Dev   | Test   | Total         |
|-----------------------|--------|-------|--------|---------------|
| <b>Sentences</b>      | 5,273  | 841   | 1,689  | <b>7,803</b>  |
| <b>Positive edges</b> | 1,995  | 457   | 1,049  | <b>3,501</b>  |
| <b>Negative edges</b> | 15,164 | 4,386 | 10,473 | <b>30,023</b> |
| <b>Total edges</b>    | 17,159 | 4,843 | 11,522 | <b>33,524</b> |

Table 5.3: Data set Statistics.

NLP tasks that are similar to variable typing have been proposed by the MIR community. Grigore et al. (2009) took operators listed in OpenMath content dictionaries (CDs) as concepts and used term clusters to model their semantics. A bag of nouns is extracted from the operator description in the dictionary and enriched manually using terms taken from online lexical resources. The cluster that maximises the similarity (based on Pointwise Mutual Information (PMI) and DICE) between nouns in the cluster and the local context of a target formula is taken to represent its meaning. Wolska et al. (2011) used the Cambridge dictionary of mathematics and the mathematics subject classification hierarchy to manually construct taxonomies used to assign meaning to simple expressions. Simple expressions are defined by the authors to be mathematical formulae taking the form of an identifier, which may have super/sub-scripted expressions of arbitrary complexity. Lexical features surrounding simple expressions are used to match the context of candidate expressions to suitable taxonomies using a combination of PMI and DICE (Wolska et al., 2011). Wolska et al. report a precision of 66%.

Schubotz et al. (2016) use hierarchical named topic clusters, referred to as namespaces, to model the semantics of mathematical identifiers. Namespaces are derived from a document collection of 22,515 Wikipedia articles. A vector-space approach is used to cluster documents into namespaces using mini-batch K-means clustering. Clusters beyond a certain purity threshold are selected and converted into namespaces by extracting phrases that assign meaning to identifiers in the selected clusters. Schubotz et al. take a ranked approach to determining the phrase that best assigns meaning to a particular identifier. The authors report  $F_1$  scores of 23.9% and 56.6% for their definition extraction methods.

Quoc et al. (2010) used a rule-based approach to extract descriptions for formulae (phrases or sentences) from surrounding context. In a similar approach, Kristianto et al. (2012) applied pattern matching on sentence parse trees and a “nearest noun” approach to extract descriptions. These rule-based methods have been shown to perform well for recall but poorly for precision (Kristianto et al., 2012). Table 5.4 shows examples of their sentential patterns for extracting descriptions. The patterns take the form of regular expressions (denoted by DEF) for target mathematical expressions (denoted by MATH). However, Kristianto et al. (2012) note that domain-agnostic parsers are confused by mathematical expressions making rule-based methods sensitive to parse tree errors. Machine learning methods were first used by Kristianto et al. (2014b) to extract descriptions for formulae. They showed that SVMs outperform rule-based description extraction methods.

| No. | Sentence Pattern  |
|-----|---|
| 1   | ...denoted(as by) MATH DEF                              |
| 2   | (let set) MATH (denote denotes be) DEF                  |
| 3   | DEF (is are)?(denoted defined given)(as by)MATH         |
| 4   | MATH (denotes denote (stand stands) for mean means) DEF |
| 5   | MATH (is are) DEF                                       |
| 6   | DEF (is are) MATH                                       |
| 7   | DEF (OTHERMATH)* MATH                                   |

Table 5.4: Sentential patterns proposed by Kristianto et al. (2012).

| No. | Sentence Pattern   |
|-----|--|
| 1   | Test whether the sentence matches one of 7 sentence patterns in Table 5.4.   |
| 2   | Test for the existence of colons, commas, or other mathematical expressions between the target mathematical expression and the definition candidate. |
| 3   | Test whether definition candidate is inside parentheses and mathematical expression is outside parentheses.  |
| 4   | Test how far definition candidates are from target mathematical expression (number of words between them).   |
| 5   | Test how closely the definition candidate is located to target mathematical expression (after or before).  |
| 6   | Surface text and POS tag of two previous and subsequent words of target mathematical expression.   |
| 7   | Surface text and POS tag of two previous and subsequent words of target mathematical expression.   |
| 8   | Apply unigram, bigram and trigram surface text and POS tag) to the beginning and end of the target mathematical expression.                          |
| 9   | Apply unigram and bigram (surface text and POS tag) to the beginning and end of the target mathematical expression.                                  |
| 10  | Surface text of verb that first appeared between the target mathematical expression and definition candidate.  |

Table 5.5: List of SVM features used by Kristianto et al. (2014) for description extraction.

The SVM proposed by Kristianto et al. (2014) used the features shown in Table 5.5. In Section 5.3.1 I propose an SVM model for variable typing that extends this set with features derived from typing edges.

Variable typing as a task for assigning denotations to elements of the symbolic modality from the textual is different to those proposed in the MIR literature in four ways:

1. Denotations (meaning) takes the form of mathematical types assigned explicitly to variables. This is in contrast to previous work by Grigore et al. (2009), Wolska et al. (2011) and Schubotz et al. (2016), who used manually constructed term clusters.
2. Unlike formula descriptions (Kristianto et al., 2012; Kristianto et al., 2014b), denotations in variable typing are assigned to constituent variables, rather than entire formulae.
3. In variable typing, variables are assigned one type from the sentence they occur in. In contrast, Grigore et al. (2009), Wolska et al. (2011) and Schubotz et al. (2016) assign

denotations globally across the corpus, while Kristianto et al. (2012; Kristianto et al. (2014b) assign descriptions to formulae from a fixed-length window around a formula.

My approach is different from that adopted in prior work because the variable typing task is specifically designed to be a prerequisite task to typed retrieval. Nevertheless, by assigning meaning to variables, rather than to more complex mathematical expressions, and by doing so at the sentential level, rather than within a fixed window, variable typing enables fine-grained control of denotation assignment:

- The meaning of complex formulae can be compositionally modelled by the collective types of their constituents.
- Type decisions at the sentential level can be collected and the meaning of variables can be refined at an arbitrary level (document and/or corpus level) using a type disambiguation algorithm (Section 6.3.1).

### 5.3 Experiments

Three new models for typing are compared to the “nearest type” baseline and the SVM proposed by Kristianto et al. (Kristianto et al., 2012; Kristianto et al., 2014b). One of my models is an extension of the latter baseline incorporating type and variable-centric features. The other two models are based on deep neural networks: a convolutional neural network and a bidirectional LSTM.<sup>4</sup>

I treat the task of typing as binary classification. An edge between a variable and a type instance is said to be *positive* if the variable is of the type in the sentence and *negative* otherwise. This means that every possible typing (= edge) in a sentence is presented to a classifier which makes a “type” or “not-type” decision.

#### 5.3.1 Models for Variable Typing

**Nearest Type baseline (NT)** In formula description extraction the nearest noun baseline defines a description as a combination of adjectives and nouns in the text preceding the target mathematical expression (Kristianto et al., 2014b). In the following example, adapted from Kristianto et al. (2014b), there are three candidate descriptions for the formula  $P(z_{ij} = t|x, z_{-ij})$  (underlined):

The posterior probability  $P(z_{ij} = t|x, z_{-ij})$  of latent topics represents the probability that a topic  $t$  is assigned to  $j^{\text{th}}$  word in  $i^{\text{th}}$  paper.

The nearest noun baseline would assign the description “the posterior probability” to the formula because it is the nearest noun to its left.

<sup>4</sup>I assume some familiarity with machine learning for my descriptions of the models in this section, but I also include a technical reference on machine learning, SVMs and relevant deep neural networks in Appendix A.2.

| <b>Type-Oriented Features</b>              |   |
|--|---|
| TypeLength                                 | Number of words in the candidate type.  |
| BaseType                                   | The base type of each candidate type.   |
| GNum                                       | The grammatical number of the type as it appears in the sentence.   |
| <b>Variable-Oriented Features</b>          |   |
| VarSym                                     | The variables and symbols in the candidate variable layout graph (one string per symbol).   |
| SymNum                                     | The number of distinct symbols in the candidate variable layout graph.  |
| BaseSym                                    | The base symbol of the candidate variable layout graph.   |
| Direction                                  | The directions (Above, Below, Up-left, Up-right, Down-left, Down-right, Next) in which a candidate symbol has neighbouring symbols. |
| Operators                                  | Operators in the mathematical context of the candidate variable layout graph.   |
| <b>Type and Variable-Oriented Features</b> |   |
| FirstLet                                   | The first letter in the type and base symbol of the candidate variable.   |
| <b>Sentence-Oriented Features</b>          |   |
| PrefSeq                                    | Prefix sequence: tokens from start of sentence to $a$ (exclusive)   |
| MidSeq                                     | Middle sequence: tokens between $a$ and $b$ (exclusive)   |
| SufSeq                                     | Suffix sequence: tokens between $b$ (exclusive) and end of sentence.  |

Table 5.6: SVM+ features. For each edge  $e$ , let  $a$  be the position of its left-most component (variable or type) and  $b$  the position of its rightmost component (variable or symbol).

In the context of variable typing, the nearest noun baseline cannot be directly computed from the task data because of my representation of the text. During the “rewriting phase” described on page 94, nouns are no longer available as atomic units as they have become parts of complex types. Instead, I approximate the nearest noun baseline using a *nearest type* baseline: given a variable  $v$ , the nearest type baseline takes the edge that minimises the word distance between  $v$  and some type in the sentence to be the positive edge. In the example above, the nearest type baseline would assign the type “posterior probability”.

**Support Vector Machine (Kristianto et al.) (SVM)** This is an implementation of the features and linear SVM described by Kristianto et al. (2014b), with hyperparameter  $C$  (the soft margin cost parameter) optimised on the development set. Due to the class imbalance in our data set I have used inversely proportional class weighting (as implemented in scikit-learn). L2-normalisation is also applied.

**Extended Support Vector Machine (SVM+)** This model is the SVM model by Kristianto et al. (2014b) with the addition of the features listed in Table 5.6. As before, I applied automatic class weighting and L2-normalisation and found that  $C = 2$  is optimal for this model through tuning over the development set.

**Convolutional Neural Network (Convnet)** Simon used a Convnet to classify each of the  $V \times T$  assignment edges as either positive or negative, where  $V$  are the variables in the input text and  $T$  are the types. Unlike in the SVM models, no hand-crafted features were used, but

| Name             | Type    | Description  |
|------------------|---------|--|
| Token            | Input   | A word in the sentence. If the token is a formula (including a variable), it is replaced by '@@@'. Types are represented by the key (type phrase in singular) of their embedding vector. |
| Token class      | Input   | An integer expressing the class of token. 0 stands for normal word, 1 for type, 2 for variable and 3 to indicate that a variable token is part of the edge being considered.             |
| Type of Interest | Input   | Class expressing the type. If the token is a type and it is part of the edge being considered, this field takes the value of 'TYPE' or '-' otherwise.                                    |
| Supertype        | Feature | Class expressing the supertype. If the token is a type, then this feature is the string key of the embedding vector of its supertype or 'NONE' otherwise.                                |

Table 5.7: Input and features to neural network typing models.

only the inputs (Table 5.7), and the pre-trained embeddings (Section 4.3).

The input is a tensor that encodes the input described in Table 5.7. The embeddings are used to represent the input tokens. In addition, two dimensions are concatenated to the input for each token: one dimension to denote whether a given token is a type (using 1 or 0) and another dimension to denote if a token is a variable.

The model has a set of different sized filters, and each filter size has an associated number of filters to be applied (all are hyperparameters to the model). The filters are applied to the input text (i.e. convolutions), and then max-pooled, flattened, concatenated, and a dropout layer ( $p = 0.5$ ) is then applied before being fed into a multilayer perceptron (MLP), with the number of hidden layers and their hidden units as hyperparameters. Finally, a softmax layer is used to output a binary decision.

The model is implemented using the Keras library<sup>5</sup> with binary cross-entropy as loss function, and the ADAM optimizer (Kingma and Ba, 2014). The aforementioned hyperparameters were tuned on the development data and balanced oversampling with replacement is used in order to adjust for the class imbalance in the data. Early stopping is used to help avoid over-fitting. The tuned hyperparameters are as follows: filter window sizes (2 to 12, then 14,16,18,20) with an associated number of filters (300 for the first five, 200 for the next four, 100 for the next three, then 75,70,50). One hidden layer of the MLP with 512 units is used with batch size 50.

**Bidirectional LSTM (BiLSTM)** The architecture (implemented by Marek Rei) takes as input a sequence of words, which are then mapped to word embeddings. For each token in the input sentence, the inputs and features described in Table 5.7 are also included.

These features are mapped to a separate embedding space and then concatenated with the word

<sup>5</sup><https://keras.io/>

embedding to form a single task-specific word representation. This allows the model to capture useful information about each word, and also designate which words to focus on when processing the sentence.

A neural sequence labeling architecture is used based on the work of Lample et al. (2016) and Rei and Yannakoudakis (2016). The constructed word representations are given as input to a bidirectional LSTM (Hochreiter and Schmidhuber, 1997a), and a context-specific representation of each word is created by concatenating the hidden representations from both directions:

$$\begin{aligned}\vec{h}_t &= LSTM(x_t, \vec{h}_{t-1}) \\ \overleftarrow{h}_t &= LSTM(x_t, \overleftarrow{h}_{t+1}) \\ h_t &= [\vec{h}_t; \overleftarrow{h}_t]\end{aligned}$$

where  $x_t$  is the word representation at step  $t$ , consisting of the token embedding and feature embeddings.

A hidden layer is added on top, in order to map the result to a more suitable vector space:

$$d_t = \tanh(W_d h_t)$$

where  $W_d$  is a weight matrix.

Finally, softmax is used as the output layer, which predicts a probability distribution over the labels ‘Type’ and ‘Not type’ for a given edge.

Rei et al. (2016) describe an extension of neural sequence labeling, where character-based word representations are combined with word embeddings using a predictive gating operation. This component was also used here, allowing the model to capture character-level patterns and estimate representations for previously unseen words. In this framework, an alternative word representation is constructed from individual characters, by mapping characters to an embedding space and processing them with a bidirectional LSTM. This representation is then combined with a regular word embedding by dynamically predicting element-wise weights for a weighted sum, allowing the model to choose for each feature whether to take the value from the word-level or character-level representation.

The model is optimised by minimising categorical cross-entropy, which is equivalent to minimising negative log-likelihood. Adadelta (Zeiler, 2012) was used for adaptively controlling learning speed, with the initial learning rate at 1.0. Words with frequency less than 2 were considered OOV, but were still used by the character-level component; all digits were replaced by 0. The word embeddings have size 100 and were initialised with word2vec (Mikolov et al., 2013b) vectors trained on the MREC document collection.

The LSTM layer size was set to 200 in each direction for both word- and character-level components; the hidden layer  $d$  was set to size 50. During training, sentences were grouped into

batches of size 64. Performance on the development set was measured at every epoch; training was stopped when performance had not improved for 10 epochs. The best-performing model on the development set was then used for evaluation on the test set.

### 5.3.2 Results

Evaluation is performed over edges, rather than sentences. I measure performance using precision, recall and  $F_1$ -score. The application of the NT, SVM, SVM+ and Convnet models on the test set yields a single run per model with 11,522 binary predictions – the total number of edges in the test set. Due to the fact that the initialisation of the BiLSTM depends on a stochastic process and is thus non-deterministic, I collect 10 runs of the model on the test set and report the mean  $F_1$ -score.

The non-parametric paired randomisation test (described in Section A.3) is used to detect significant differences in performance across classifiers. Comparisons using the randomisation test that involve the BiLSTM are executed using the run that performed the closest to the mean  $F_1$  score over the 10 runs of the model. The Wilcoxon rank-sum test is used to detect differences in the distribution of certain features of the data points.

Table 5.8 shows the performance results of all classifiers considered. All three proposed models have significantly outperformed the NT baseline and the SVM by Kristianto et al. (2012) and Kristianto et al. (2014b), which represents the state-of-the-art machine learning method for assigning descriptions to formulae at the time of writing. The best performing model is the bidirectional LSTM ( $F_1 = 78.98\%$ ), which significantly outperformed all other models ( $\alpha = 0.01$ ).

|                | Precision (%) | Recall (%) | $F_1$ -score (%) |
|----------------|---------------|------------|------------------|
| <b>NT</b>      | 30.30         | 82.94      | 44.39            |
| <b>SVM</b>     | 55.39         | 76.36      | 64.21            |
| <b>SVM+</b>    | 71.11         | 72.74      | 71.91            |
| <b>Convnet</b> | 80.11         | 70.26      | 74.86            |
| <b>BiLSTM</b>  | 83.11         | 74.77      | 78.98            |

Table 5.8: Model performance summary. Differences between all pairs are statistically significant ( $p < 0.01$ ) according to the permutation test.

According to the results in Table 5.8, both deep neural network models have significantly outperformed classifiers based on other paradigms. This is consistent with the intuition that the language of mathematics is formulaic: we expect deep neural networks to effectively recognise patterns and identify correlations between tokens. It is interesting to note that the Convnet model outperforms SVM+, given that the construction of the latter model involved manually engineered features. In contrast, no manual feature engineering has been performed on the Convnet model (or indeed on any of the deep neural network models). This implies that the learned type embeddings have been conducive for these models.

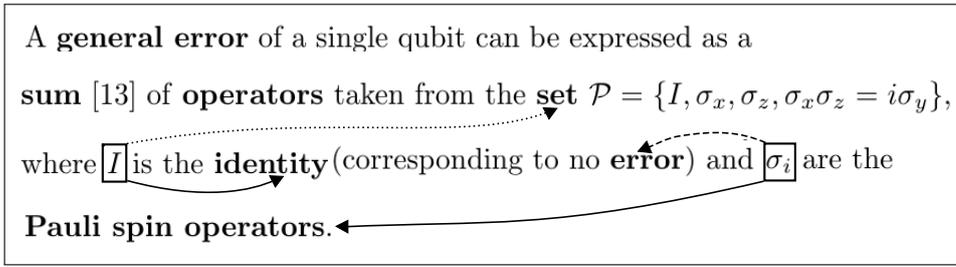


Figure 5.2: Two typings and possible false positive links.

The nearest type (NT) baseline demonstrates high recall but low precision. This is not surprising since the NT baseline is not capable of making a negative decision: it always assigns some type to all variables in a given sentence. According to my assumption 1, however, if an instance of a variable is typed outside the current sentence, the correct answer in these cases is “Type Unknown”, an option not available to this baseline.

|                | Successes | Failures |
|----------------|-----------|----------|
| <b>SVM</b>     | 25        | 379      |
| <b>SVM+</b>    | 12        | 74       |
| <b>Convnet</b> | 31        | 104      |
| <b>BiLSTM</b>  | 46        | 57       |

Table 5.9: Unique success/failure count per model.

Table 5.9 lists numbers of unique successes and failures (instances where a model is the only model to make the correct or wrong prediction respectively) for the four sophisticated models<sup>6</sup>. The SVM model by Kristianto et al. produces a high number of unique failures. Despite the fact that the SVM+ model has half the number of unique successes of the SVM model, it has considerably fewer unique failures (74 as opposed to 379). The BiLSTM achieves the largest number of unique successes and the least number of unique failures (46 and 57 respectively).

The performance of the models could also be affected by long source sentences and long edge length (distance between a variables and its types). Figure 5.2 illustrates this using an example from ArXiv quant-ph/9611027. Solid lines in the figure represent positive edges between variables (boxed tokens) and types (bold text); dotted and dashed lines represent false positives. There is a clear positive link between variable  $I$  and the type “identity”: the two are very close and linked via the copular verb “is”. This edge is typical for mathematical text and it is not surprising that it has been identified by all models considered. However, both the SVM and SVM+ model additionally identified the false positive (dotted line) between  $I$  and “set” (remember that there is no formal enforcement of the uniqueness constraint between variables and types) – it appears that the SVM models cannot recognise the fact that  $I$  is part of an adverbial phrase and that in this context, valid edges should not link outside of the phrase.

The positive edge between  $\sigma_i$  and type “Pauli spin operators” has only been correctly identified

<sup>6</sup>The Nearest Type baseline NT is excluded due to the fact it always assigns a type.

by the deep neural network models, although the verb “are” occurs between the edge points. The failure of the other models to find this link may be attributable to the fact that the sentence is by that point quite long and structurally complex. NT and SVM generated the false positive edge between  $\sigma_i$  and “error” (dashed line), seemingly misled by the close proximity of the variable to the (wrong) candidate type.

Looking into distribution location differences in terms of mean and median in sentence length between unique successes (according to the permutation test for independent samples, (Siegel and Castellan, 1988)) I found no statistically significant differences between model pairs. In contrast, I found significant (at  $\alpha = 0.05$ ) differences between the average sentence length in the pair SVM/BiLSTM (32.7 vs 38.3). In terms of differences in median, there were significant differences in the pairs SVM/ConvNet (median length of sentence 29 vs 34 words), SVM/BiLSTM (29 vs 36 words), SVM+/ConvNet (28 vs 34 words) and SVM+/BiLSTM (28 vs 36).

Looking into the distribution of unique successes in terms of edge length (distance between variable and type), I found no significant differences in any pair of models. However, I found significant differences ( $\alpha = 0.05$ ) in mean between the distributions of failures for the pairs SVM/ConvNet (average edge length 3.4 vs 8.4), SVM/BiLSTM (mean 3.4 vs 6.2 respectively) and SVM+/ConvNet (3 vs 8.4). In terms of median, I found significant differences in the pairs SVM/SVM+ (median edge length 3 vs 1), SVM/ConvNet (3 vs 4.5), SVM+/ConvNet (1 vs 4.5) and SVM+/BiLSTM (1 vs 3).

This investigation into successes and failures suggests that variable typing models based on neural networks are more robust to long sentences and edges than those based on SVMs. However, I cannot draw definitive conclusions since the parameters learned by the neural network models have no intuitive interpretation.

## 5.4 Variable Typing for Typed Retrieval

I designed variable typing as a standalone NLP task. One design decision for the variable typing task has been to only consider simple variables – a variable definition adapted from Grigore et al. (2009) – for edges with types in the text, because this is the established variable definition in the MIR community.

However, typing and indexing simple variables in this manner is restrictive because it overlooks phenomena where simple variables are declared using complex expressions, such as declarations of maps or assignments. To illustrate, consider the following definition from Fraleigh (2003):

**Definition 0.12:** A function  $\phi : X \rightarrow Y$  is one to one if  $\phi(x_1) = \phi(x_2)$  only when  $x_1 = x_2$ ...The function  $\phi$  is onto  $Y$  if the range of  $\phi$  is  $Y$ .

The formula  $\phi : X \rightarrow Y$  (map notation) does not fit the definition of “simple variable” by Grigore et al. (2009) and in the example above, the formula partakes in a type assignment:

$\phi : X \rightarrow Y$  is expected to be of type “function”. However,  $\phi : X \rightarrow Y$  is not a simple variable and variable typing will not consider the edge between it and the type “function”.

In the first sentence of this example, the variable  $\phi$  is declared to be a “function” using map notation but is subsequently referred to using function notation:  $\phi(x_1)$ . In the second sentence, the function is referred to using only its symbol,  $\phi$ . The formulae  $\phi(x_1)$  and  $\phi$  are anaphoric; they reference the original declaration of the variable  $\phi$ , which informs the reader about the domain and range of the function and is thus more detailed. This phenomenon is prolific in mathematical literature and can be observed in many branches of mathematics.

My goal is to expand the number of variables in the test collection that can be typed and to increase the recall of typings that can inform typed retrieval. I do this by introducing two pieces of machinery built on top of variable typing to propagate type information from declarations to anaphoric variables in the discourse.

First, I extend the definition of “variable” to distinguish between symbol *declarations* and *anaphoric* instances by introducing the notion of *head variable*. A head variable is a simple variable that is used to reference a complex formula or declaration and is always first introduced in the formula or declaration it references. The variable  $\phi$  in the example above, for instance, is the head variable of the map declaration  $\phi : X \rightarrow Y$ . I have identified 8 patterns from which head variables can be extracted. These patterns are shown in Table 5.10.

| Pattern Example             | Head                | Description                                 |
|-----------------------------|---------------------|---|
| $X_{expr2}^{expr1}$         | $X_{expr2}^{expr1}$ | Variables with arbitrarily complex scripts. |
| $x = expr$                  | $x$                 | Instance of an assignment or identity.      |
| $f(x) = expr$               | $f$                 |   |
| $x \in S$                   | $x$                 | Instance of membership.                     |
| $c : I \rightarrow G$       | $c$                 | Declaration of a map.                       |
| $ D $                       | $ D $               | Fenced variable.                            |
| $f(x)$                      | $f$                 | A function call for some parameters.        |
| $F ::= 0 F \wedge F  \dots$ | $F$                 | Context-free grammar declaration.           |
| $x \approx 3$               | $x$                 | Approximate value.                          |

Table 5.10: Special classes of variables recognised by my variable parser.

The second piece of machinery is a generator that expands the number of candidate edges for variable typing. This generator

1. extends the variable parser used to detect simple variables in variable typing to emit the head variable of a formula. If a head variable for a formula can be found, it is used to generate variable typing candidate edges; and then
2. redirects the type assignment induced by a typing  $(F_1, T_1)$ , where  $F_1$  is a formula and  $T_1$  a type, to the head variable of  $F_1$ .

Variable typing for typed retrieval now works as follows. During the candidate generation step of my algorithm, the variable parser also emits the head variable of formulae that match the patterns in Table 5.10. Head variables are used to generate candidate edges that would have otherwise not be considered in variable typing. I use the best variable typing model found in my earlier experiments, the BiLSTM model, to classify candidate edges.

There are two advantages to applying this generator for typed retrieval. First, it allows me to extract typings for formulae matching the patterns in Table 5.10 that would have otherwise been ignored because they are not simple variables. For example, in Definition 0.12 above, a typing ( $\phi : X \rightarrow Y$ , “function”) is now discoverable. Extracting types from declarations is important because, as shown by Wolska and Grigore (2010), the meaning of the majority of symbols is given in the sentence in which they are declared. The second advantage concerns the propagation of the type of declarations to subsequent instances in the same document. For example, upon encountering Definition 0.12 above, typed retrieval models will know that instances of the variables  $\phi$  and  $\phi(x_1)$  are references to  $\phi : X \rightarrow Y$  and have type “function”. Recognising the types for these instances would not have been possible without my extensions.

My generator does not alter the assumptions or operation of variable typing. Rather, it conceptually transforms declarations to semantically equivalent ones that can be typed by a variable typing model trained on the data set presented in this chapter. For example, the declarations

A function  $\phi : X \rightarrow Y$  is one to one if  $\phi(x_1) = \phi(x_2)$  only when  $x_1 = x_2$ .

A function  $\phi$  is one to one if  $\phi(x_1) = \phi(x_2)$  only when  $x_1 = x_2$ .

are semantically equivalent. The only difference is that typings in the latter can be recognised by variable typing whereas typings in the former cannot be recognised without the aforementioned extensions.

## 5.5 Limitations of Variable Typing

My variable typing machinery admittedly has some limitations in the context of typed retrieval. The first limitation is that variable typing is a machine learning task and in some cases the wrong type can be assigned to a variable (i.e., a false positive). Some false positive type assignments might get overridden by decisions made at the type disambiguation stage of typed retrieval (section 6.3.1) if the correct type for a particular variable can be extracted somewhere in the source document. Like any other process based on machine learning, however, typed retrieval is subject to noise coming from false positives.

Another limitation is that the list of patterns in Table 5.10 is not exhaustive. I produced this list empirically by observing discussions on MathOverflow (threads outside of my test collection) and mathematical text in ArXiv papers. Nevertheless, it should capture many instances of

variables that would have otherwise been ignored during variable typing that can now be used in typed retrieval.

A third limitation of my approach is that references to mathematical objects using syntactically ambiguous formulae cannot be typed. For example, consider the following question on MathOverflow<sup>7</sup>:

The *blowdown* of the *zero section* of the *canonical bundle* of the *first del Pezzo surface*  $dP_1$ , the *blowup* of  $CP^2$  at one *point*, is a *Calabi-Yau cone*... Is the same thing true for the *blowdown* of the *zero section* of the *canonical bundle* of the *second del Pezzo surface*  $dP_2$ ...

In the example above, a human reader can easily deduce that  $dP_1$  :: “first del Pezzo Surface” and  $dP_2$  :: “second del Pezzo Surface” are distinct variables. However, when this task is done automatically, it is not unambiguously clear from the Presentation MathML whether  $dP$  is a single identifier or two adjacent, yet distinct symbols  $d$  and  $P$  separated by an implicit binary operator. As a result,  $dP_1$  and  $dP_2$  cannot be recognised as complex variables at all, and therefore all processing breaks down here.

I have chosen not to attempt to correct such instances because doing so might affect the intended meaning of formulae. For example, one could imagine an instance where an implicit binary operator exists between  $d$  and  $P$ . In this case, merging the two symbols into a single SLT identifier would result in the meaning of SLT to deviate from that of the source formula.

## 5.6 Chapter Summary

In this chapter I introduced the variable typing task as a stand-alone NLP task (up to Section 5.3). My variable typing task differs from previous work in three important ways: (a) denotation (meaning) in the form of mathematical types is explicitly assigned to variables, rather than arbitrary mathematical expressions, (b) variables are assigned those denotations that are present in the same sentence they occur (rather than being drawn from a context of arbitrary length, e.g. fixed windows or documents) and (c) denotations are drawn from a pre-determined list of labels, rather than free-form text from the context surrounding each variable.

With the help of Simon Baker, I have constructed a new data set for the variable typing task. My data set, which is publicly available, contains 33,524 labelled edges (positive or negative) in 7,803 sentences. Simon Baker and Marek Rei helped me build and evaluate three variable typing models. Our models outperform the current state-of-the-art methods developed for similar tasks. Marek Rei’s BiLSTM model is the top performing model for variable typing, achieving 79%  $F_1$ -score.

In the context of this thesis, variable typing is a prerequisite machine learning task for typed retrieval – I use the best variable typing model (Marek’s BiLSTM model) as a pre-processing

---

<sup>7</sup>MathOverflow thread 12962, <https://mathoverflow.net/questions/12962/existence-of-smoothing-of-calabi-yau-cones-over-dp-1-and-dp-2>.

step to assign types to variables in the formulae of documents and queries. I also for practical reasons extended the scope of variables that can be assigned a type by the task, without loss of generality in section 5.4. My expectation, as suggested in more detail at the beginning of chapter 4, is that typed formula retrieval will improve retrieval efficiency by reducing the number of false positive associations between mathematical expressions in query and document. I will discuss retrieval models and experiments based on typed retrieval in more detail in chapters 6 and 7 respectively.



# Chapter 6

## Typed Joint Retrieval

In Chapter 4 we have seen that types are a natural part of the mathematical text and in Chapter 5 I described how types can be used to link the textual and symbolic modalities. In this chapter I will discuss how MIR models can use the link established through variable typing to jointly model formulae and text for MIR retrieval.

Most MIR systems treat text and formulae as independent sources of information. Indexing and retrieval of the symbolic modality is done independently of the textual modality and processing the former modality does not use information coming from processing the latter modality when modelling or retrieving mathematical information. In Tangent (Pattaniyil and Zanibbi (2014), section 2.2.5), for example, indexes and scores formulae using triples of symbols while text is processed using the traditional retrieval model VSM. Two independent scores for each modality are produced by Tangent and combined into a single score using weighted linear interpolation.

I propose linking the symbolic and textual modalities by replacing each variable in formulae with one type extracted from the textual modality. The main idea behind this form of typed retrieval is that finding similar formulae in queries and documents becomes more flexible when formulae are matched based on the types of their constituents, rather than their raw symbols. This takes the form of tree matching, as formulae are tree structures. Ultimately, this chapter is concerned with the algorithms that realise typed unification: algorithms for typing formula trees and their constituents and algorithms for matching the typed tree structures in documents with those in the queries.

In order to illustrate the proposition that types introduce a dependence between text and formulae in the discourse, I will now develop an example. Suppose that a mathematician is looking for a specific form of homomorphism. The searcher knows that the homomorphism she is looking for involves instances of groups but is not sure which exactly. In her query, she writes the formula

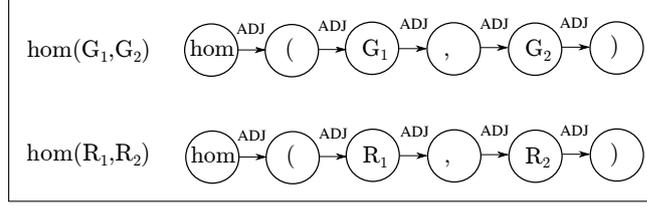
$$\text{hom}(G_1, G_2) \tag{1}$$

and the textual component of her query specifies that  $G_1 :: \text{GROUP}$  and  $G_2 :: \text{GROUP}$ .

Suppose that in reality, the homomorphism that will satisfy her need is:

$$\text{hom}(R_1, R_2) \tag{2}$$

where  $R_1 :: RING$  and  $R_2 :: RING$ .



| $\text{hom}(G_1, G_2)$ |       |       |       | $\text{hom}(R_1, R_2)$ |       |       |       |
|------------------------|-------|-------|-------|------------------------|-------|-------|-------|
| Parent                 | Child | Dist. | Vert. | Parent                 | Child | Dist. | Vert. |
| hom                    | (     | 1     | 0     | hom                    | (     | 1     | 0     |
| hom                    | $G_1$ | 2     | 0     | hom                    | $R_1$ | 2     | 0     |
| hom                    | ,     | 3     | 0     | hom                    | ,     | 3     | 0     |
| hom                    | $G_2$ | 4     | 0     | hom                    | $R_2$ | 4     | 0     |
| hom                    | )     | 5     | 0     | hom                    | )     | 5     | 0     |
| (                      | $G_1$ | 1     | 0     | (                      | $R_1$ | 1     | 0     |
| (                      | ,     | 2     | 0     | (                      | ,     | 2     | 0     |
| (                      | $G_2$ | 3     | 0     | (                      | $R_2$ | 3     | 0     |
| (                      | )     | 4     | 0     | (                      | )     | 4     | 0     |
| $G_1$                  | ,     | 1     | 0     | $R_1$                  | ,     | 1     | 0     |
| $G_1$                  | $G_2$ | 2     | 0     | $R_1$                  | $R_2$ | 2     | 0     |
| $G_1$                  | )     | 3     | 0     | $R_1$                  | )     | 3     | 0     |
| ,                      | $G_2$ | 1     | 0     | ,                      | $R_2$ | 1     | 0     |
| ,                      | )     | 2     | 0     | ,                      | )     | 2     | 0     |
| $G_2$                  | )     | 1     | 0     | $R_2$                  | )     | 1     | 0     |
| )                      | None  | 0     | 0     | )                      | None  | 0     | 0     |

Figure 6.1: Top: Untyped SLTs for the formulae  $\text{hom}(G_1, G_2)$  and  $\text{hom}(R_1, R_2)$ , bottom: Tangent formula index entries for  $\text{hom}(G_1, G_2)$  and  $\text{hom}(R_1, R_2)$ .

For the formulae  $\text{hom}(G_1, G_2)$  and  $\text{hom}(R_1, R_2)$ , Tangent (Pattaniyil and Zanibbi, 2014) will use the SLTs in the top of Figure 6.1 to produce the index tuples shown in the bottom of the figure<sup>1</sup>. The sets of Tangent index tuples entries for the two formulae are almost identical. However, because the formulae use different symbols at the same positions, the tuples for formula 1 are not exact matches to the tuples for formula 2 and the Tangent scores will not reflect their true similarity.

One of the things that Tangent (Pattaniyil and Zanibbi, 2014) misses is that Formulae 1 and 2 are syntactically equivalent, since the substitutions  $\{G_1 \rightarrow R_1, G_2 \rightarrow R_2\}$  can be used to turn formula 1 into 2. Note that syntactic equivalence is another name for  $\alpha$ -equivalence, which was first introduced on page 27 and which we will return to in section 6.4.1.

Any MIR system can take advantage of syntactic equivalence to match and score formulae.

<sup>1</sup>I am simplifying the SLTs and index by including the subscripts in the identifier nodes.

However, a typed retrieval system can take this a step further: it will create the typed SLTs in Figure 6.2 (right) instead of the corresponding untyped SLTs (left) and determine that the type of  $G_1$  can be coerced to the type of  $R_1$  and that the type of  $G_2$  can be coerced to the type of  $R_2$  by deducing that a *RING* is a specialisation of *GROUP*.

| Formula                | Untyped SLT | Typed SLT |
|------------------------|-------------|-----------|
| $\text{hom}(G_1, G_2)$ |             |           |
| $\text{hom}(R_1, R_2)$ |             |           |

Figure 6.2: Untyped and typed SLTs for the formulae  $\text{hom}(G_1, G_2)$  and  $\text{hom}(R_1, R_2)$ .

This mechanism, which I call type unification (Section 6.2), happens during scoring and enables typed retrieval to recognise that formula 2 is not only syntactically but also semantically relevant to formula 1. This leads to a successful match in this case. I borrow ideas for this mechanism from the field of denotational semantics for programming languages and I refer to matching and scoring based on this mechanism as “semantic” with this in mind.

Typed retrieval is not always useful. In some cases, mathematical enquiries can be expressed without any typable expressions. The following MathOverflow question<sup>2</sup> is an example:

Is there any classification for coadjoint orbits of lower or upper triangular matrices in general case  $n \times n$ . Is there any reference?

Despite the fact that the question above contains two types, the only mathematical expression in the text,  $n \times n$ , is not typable since  $n$  is neither typed nor declared in the question’s body. In other cases, mathematical enquiries can be expressed with no types in the text at all<sup>3</sup>:

How can I find the general solution of  $a^4 + b^4 = c^4 + d^4$  ( $a, b, c, d > 0$ )? And how did Euler find the solution  $158^4 + 59^4 = 133^4 = 134^4$ ?

In both of these cases, it is of course impossible for typed retrieval to make any difference. However, in the vast majority of cases types are present both in language and in the formulae, and the approaches in this chapter explore how this can be exploited.

In the sections that follow I will describe the individual components of typed retrieval in more detail and explain my overall approach at specialising them. Then, in Section 6.4 I will introduce concrete models that use the ideas in this chapter to realise typed retrieval.

<sup>2</sup>MathOverflow thread 127010, <https://mathoverflow.net/questions/127010/classification-for-coadjoint-orbits-of-lower-or-upper-triangular-matrices>.

<sup>3</sup>MathOverflow thread 142434, <https://mathoverflow.net/questions/142434/diophantine-equation-a4b4-c4d4-a-b-c-d-0>.

## 6.1 Matching Operations on SLTs

I use three different types of matching: exact matching, syntactic and recursive matching, which are realised by matching operators. In order to represent formulae, I need a tree-like structure; out of all possible such structures, I adopt and extend SLTs (section 6.1.1).

Exact tree matching addresses the case where two SLTs are identical in structure as well as in their vocabulary of symbols. This happens often when a document or query has many occurrences of a particular symbol or identifier, or in the general case, if an identifier is short. However, for more complex formulae, matching SLTs with raw symbols directly is counter-productive: it is unlikely that two equivalent expressions will be written down using the exact same collection of symbols. Syntactic matching addresses this problem by first performing syntactic normalisation (also known as skolemisation), thereby enforcing  $\alpha$ -equivalence, which makes the matching of identically structured formulae more flexible.

Beyond  $\alpha$ -equivalence, there is one more piece of information that can increase our certainty that two non-identical formulae can be semantically related, and that is type unification. If we can (a) determine that two formulae have the same structure using  $\alpha$ -equivalent matching and (b) have symbols with *unifiable* types at the same positions in their SLTs, then we can be reasonably certain that a document formula is relevant to a query formula. In Section 6.2 I will introduce type unification which makes this happen.

The operations discussed in this section are used in one form or another in all components of typed retrieval:

1. Determining whether an SLT represents a simple or complex variable uses the `exact` operator when matching the SLT against the patterns in Table 5.10 from Section 5.4.
2. The matching operators are also used during type disambiguation (to be discussed in section 6.3.1), when equivalence classes of formulae are compiled (candidate types are collected for each class from the document or query).
3. Tree matching (to be discussed in sections 6.4.1 and 6.4.2) uses exact, syntactic and recursive matching to produce the information needed to score query–document formula pairs.

### 6.1.1 Extensions to SLTs

Compared to the SLTs in the literature I made a few extensions for practical reasons: my extensions implement more informative mathematical knowledge and thus allow for earlier recognition of non-matching parts of formulae. My SLTs (a) encode more layout relationships between constituents and (b) have a larger dictionary of non-terminals, leading to more fine-grained representations of formula structure thus allowing for more structural idioms to be modelled explicitly.

As do Zanibbi and Yuan (2011), Schellenberg et al. (2012) and Pattaniyil and Zanibbi (2014), I also parse Presentation MathML into symbol layout trees (SLTs), and produce SLTs for all

formulae in a document or query. The reason for this is that Presentation MathML can be produced from  $\text{\LaTeX}$  more reliably than Content MathML at scale.

In contrast to Pattaniyil and Zanibbi’s 5 kinds of SLT edges<sup>4</sup>, my SLTs are connected in 7 directions<sup>5</sup>: ABOVE, BELOW, SUBSCRIPT, SUPERSCRIPPT, LEFT SUBSCRIPT, LEFT SUPERSCRIPPT and NEXT positions. Every SLT node (an instance of `LayoutNode`) can also have a WITHIN subtree to encode constructs like square roots. These relationships are illustrated in Figure 6.3.

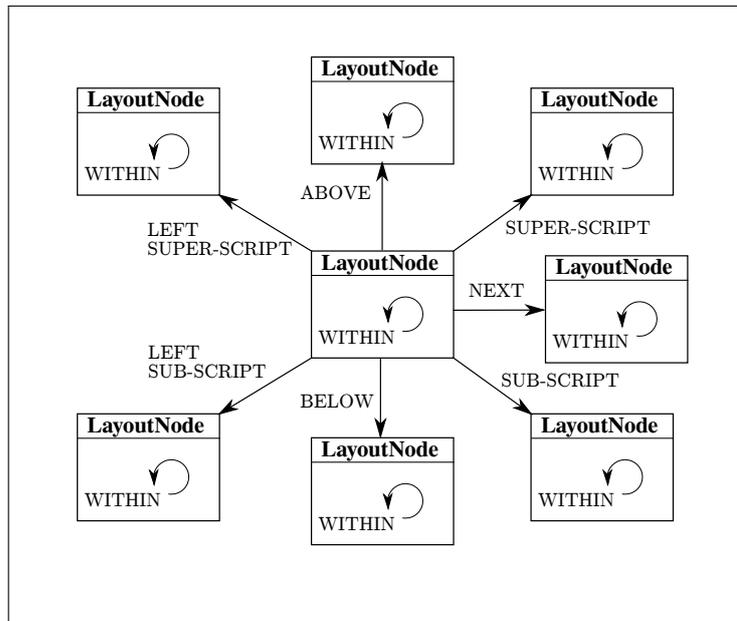


Figure 6.3: Structural relationships between layout nodes in my SLTs.

The next point of difference concerns the non-terminals, which correspond to structural patterns or idioms. Non-terminals as idioms encapsulate patterns in formulae that convey specific mathematical meaning through their layout. For example, Tangent and other MIR systems have a special SLT node for fractions. This node has two children: one node in the ABOVE direction and another in the BELOW direction. Figure 6.4 illustrates this by showing Tangent’s representation for the formula  $\frac{x^2}{\sqrt{z}}$ . We can see that the non-terminal SLT node `Sqrt` is the “BELOW” child of the formula, with a further terminal node of type constant as a “WITHIN” child.

<sup>4</sup>Namely, ADJ (adjacent), ABOVE, BELOW, SUPERSCRIPPT and WITHIN.

<sup>5</sup>There is some uncertainty about who invented 7-directional SLTs. At the time I conducted my experiments, nobody had published any of these extensions (and I had not either), but since then, 7-directional SLTs have been introduced in the literature, without any statement of novelty. I am reporting this fact only for completeness, because in my mind these extensions are a natural iteration on SLTs, and as such prone to simultaneous invention.

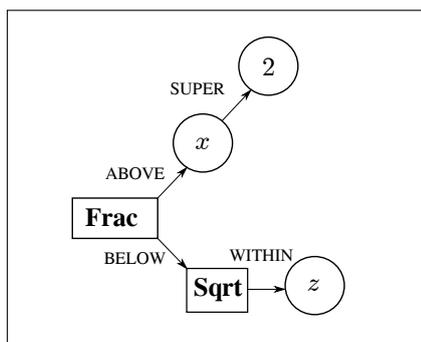


Figure 6.4: Target SLT for  $\frac{x^2}{\sqrt{z}}$ .

The advantage of such a representation is that it allows for abstraction over similar fractions, such as  $\frac{x^3}{5}$ , which has a similar structure to  $\frac{x^2}{\sqrt{z}}$ , both of which share structurally similar ABOVE parts. The BELOW parts are structurally different, and Tangent’s representation makes it possible to identify quickly that the former has a constant where the latter has a non-terminal (namely the square root non-terminal).

The non-terminals used in SLTs I have seen in the literature enable a certain level of generalisation. However, my hypothesis is that the vocabulary of non-terminal nodes could be expanded to capture more structural idioms leading to better generalisation and thus better matching.

My first observation was that it is unclear exactly how many such non-terminal nodes are in use, as no definitive list is published in the SLT literature. From the few published examples I had access to, I could already see that additional non-terminal nodes are possible which should, in my opinion, help generalisation. This led to my design of an improved SLT representation that encodes more structural patterns and idioms in its non-terminals than are currently in use.

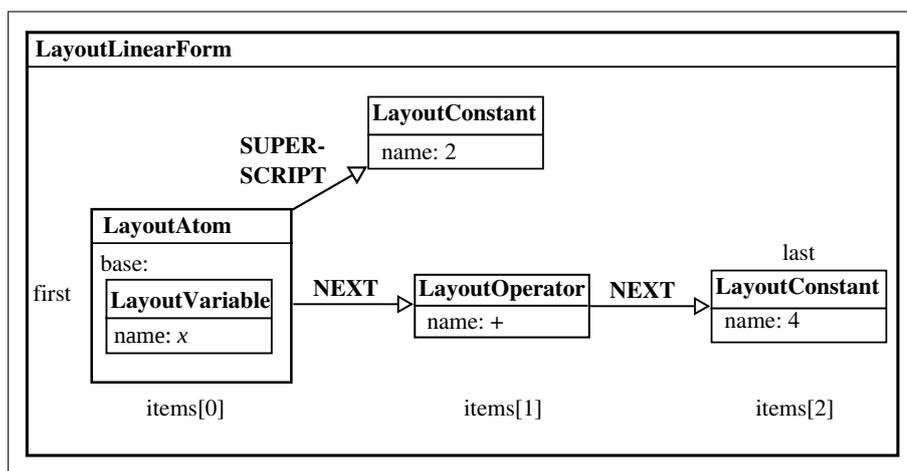


Figure 6.5: Representation example  $x^2 + 4$ : 7-directional SLTs.

In my SLT representation, I defined nine structural idioms (non-terminal nodes):

- **LayoutLinearForm**: This non-terminal is used to represent mathematical expressions with constituents laid out on the same typographical baseline. Consider, for example the

SLT representation of  $x^2 + 4$  in Figure 6.5. This SLT in Figure 6.5 encapsulates three nodes aligned on the same typographical baseline that represent the sub-expression  $x^2$ , the operator  $+$  and the numerical constant “4” in sequence.

- **LayoutAtom:** The `LayoutAtom` non-terminal is exclusively used to represent scripting between a base expression. The expression  $x^2$  in Figure 6.5, for example, is represented as a `LayoutAtom` with a `LayoutVariable` terminal as the base and a `LayoutConstant` terminal as its super-scripted sub-expression.
- **LayoutAboveBelow:** This non-terminal is used to represent expressions where constituents are laid out on top of each other. Typical examples include binomial coefficients and fractions.
- **LayoutFraction:** This non-terminal is a special instance of `LayoutAboveBelow` used to represent fractions. I use a special non-terminal for fractions due to their frequency in mathematical text.
- **LayoutEnclosure:** This non-terminal is used to represent expressions that involve one expression enclosing another.
- **LayoutRoot:** A special instance of the non-terminal `LayoutEnclosure` used to represent the  $n^{\text{th}}$  root of an enclosed expression. This non-terminal allows for the specification of a base (typically typeset as an upper-left superscript) stored as a child SLT. I created a specialised non-terminal for roots due to their high frequency of occurrence in mathematical text.
- **LayoutGrid:** This node type represents any formulae that might be organised as a 2D orthogonal grid. Each row is represented by an instance of the non-terminal `LayoutLinearForm`.
- **LayoutMatrix:** A special instance of `LayoutGrid` used to represent matrices which also contains the dimensions of the encoded matrix. The dimensions of a matrix are used by Tangent for indexing matrix structures (Pattaniyil and Zanibbi, 2014) and I replicate this behavior in my implementation of Tangent indexing.
- **LayoutTable:** A special instance of `LayoutGrid` used to represent grids that are not matrices.

As will become clearer in the next sections, a larger vocabulary of non-terminals should make SLTs more precise and more expressive about shallow semantics. This leads to earlier rejection of spurious matches between formulae.

### 6.1.2 Exact SLT Matching

The exact matching SLT operation, which I refer to as  $\text{exact}(SLT_i, SLT_j)$ , realises equality of formula SLTs,  $SLT_i =_{SLT} SLT_j$ . The operation traverses  $SLT_i$  and  $SLT_j$  simultaneously in a depth-first manner. At each step, it checks whether the terminal (e.g., constants and literals) and non-terminal nodes (e.g., structural idioms) are the same at isomorphic positions on the two SLTs. If the nodes differ at the same position, then  $\text{exact}$  will report that the SLTs are not the same. If, on the other-hand,  $\text{exact}$  matches all sub-trees of the SLTs then it will report they are identical.

I will now illustrate the operation of  $\text{exact}$  using some concrete examples. Suppose that  $\text{exact}$  is to match the identical expressions  $F_1 = \frac{x^2}{\sqrt{z}}$  and  $F_2 = \frac{x^2}{\sqrt{z}}$ . The SLTs for these expressions are shown in Figure 6.6. The steps taken by  $\text{exact}$  to determine that the SLTs for  $F_1$  and  $F_2$  are exact matches are shown in Table 6.7. The steps in Figure 6.7 demonstrate that  $\text{exact}$  determines that all sub-trees of the SLTs in isomorphic positions are exact matches, making the two SLTs equivalent.

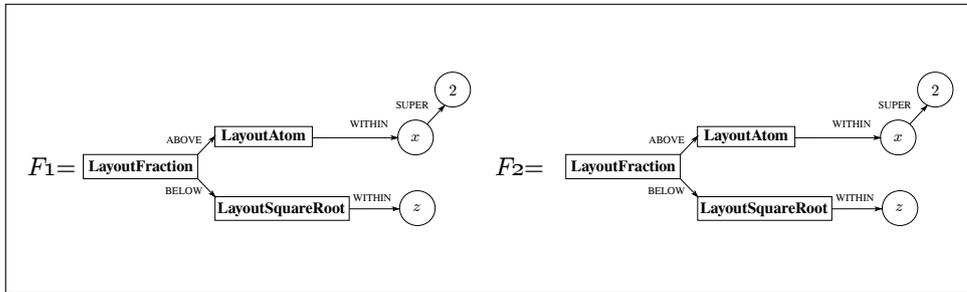


Figure 6.6: Identical SLTs for  $F_1$  and  $F_2$  ( $\frac{x^2}{\sqrt{z}}$ ).

| Step   | Direction       | Result |
|--|-----------------|--------|
| $\text{exact}(\text{LayoutFraction}, \text{LayoutFraction})$     | ROOT            | True   |
| $\text{exact}(\text{LayoutSquareRoot}, \text{LayoutSquareRoot})$ | BELOW           | True   |
| $\text{exact}(\text{LayoutLiteral}, \text{LayoutLiteral})$       | WITHIN          | True   |
| $\text{exact}(z, z)$   | Literal values  | True   |
| $\text{exact}(\text{LayoutAtom}, \text{LayoutAtom})$             | ABOVE           | True   |
| $\text{exact}(\text{LayoutLiteral}, \text{LayoutLiteral})$       | WITHIN          | True   |
| $\text{exact}(x, x)$   | Literal values  | True   |
| $\text{exact}(\text{LayoutLiteral}, \text{LayoutLiteral})$       | SUPER           | True   |
| $\text{exact}(2, 2)$   | Constant values | True   |

Figure 6.7: Run for  $\text{exact}$  on the SLTs of  $F_1$  and  $F_2$ .

I will now illustrate the case where two SLTs are not equivalent. Consider the two expressions  $F_1 = \frac{x^2}{\sqrt{z}}$  and  $F_2 = \frac{x^2+y}{\sqrt{z}}$ , their SLTs, shown in Figure 6.8.

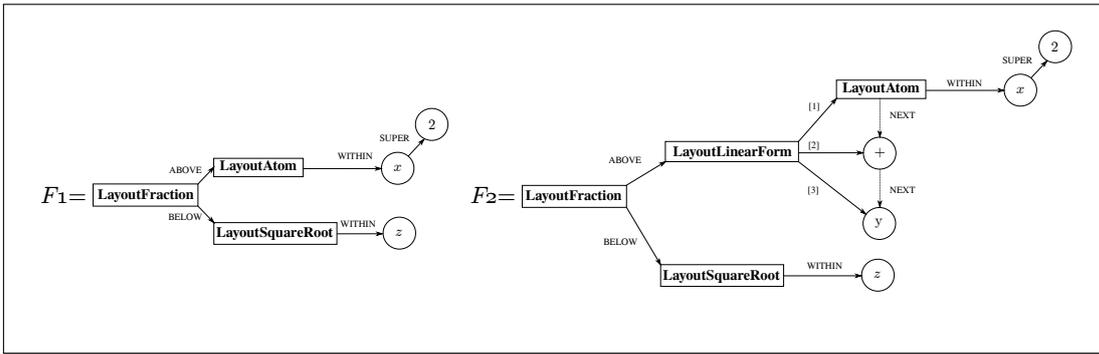


Figure 6.8: Non-equivalent SLTs for the distinct formulae  $F_1 = \frac{x^2}{\sqrt{z}}$  and  $F_2 = \frac{x^2+y}{\sqrt{z}}$ .

As shown in Figure 6.9, `exact` matches the `BELOW` branches of  $F_1$  and  $F_2$ , but in the last step it recognises that the non-terminals being compared are not identical in the `ABOVE` direction ( $F_1$  has a `LayoutAtom`, while  $F_2$  has a `LayoutLinearForm`) and returns `False`.

This example also shows the usefulness of my extensions to the SLT non-terminals. If I had used the SLTs commonly used in the literature, the `LinearForm` on the right would not be there; instead, the expression  $y + x^2$  would take the form of a deep branch, with the node for  $y$  being the root connecting the other symbols one edge at a time. The entirety of this branch would have to be searched and a non-match would be found at a much later state. My algorithm allows for the no-match to be caught at an early stage, namely when the `LinearForm` is encountered during the tree walk.

| Step  | Direction      | Result |
|---|----------------|--------|
| <code>exact (LayoutFraction, LayoutFraction)</code>     | ROOT           | True   |
| <code>exact (LayoutSquareRoot, LayoutSquareRoot)</code> | BELOW          | True   |
| <code>exact (LayoutLiteral, LayoutLiteral)</code>       | WITHIN         | True   |
| <code>exact (z, z)</code>                               | Literal values | True   |
| <code>exact (LayoutAtom, LayoutLinearForm)</code>       | ABOVE          | False  |

Figure 6.9: `exact` matching on the SLTs of  $F_1 = \frac{x^2}{\sqrt{z}}$  and  $F_2 = \frac{x^2+y}{\sqrt{z}}$ .

I perform matching under the assumption of implicit commutativity, i.e., the assumption that all expressions and operators arranged in the same typographical baseline in Presentation MathML are commutative. The algorithm that enforces implicit commutativity is incorporated in `exact` for the case where both inputs are of non-terminal kind `LayoutLinearForm`. All children of the two `LayoutLinearForm` non-terminals are placed in a table that enumerates all possible pairings between the sub-expressions of the two non-terminal nodes. Then, `exact` is applied to each pair and the `LayoutLinearForm` nodes are considered to be a match if the number of matching sub-expressions pairs is equal to the number of children in the `LayoutLinearForm`

non-terminals.

To illustrate how implicit commutativity is enforced by `exact`, I will use the example  $F_1 = x^2 + 3$  and  $F_2 = 3 + x^2$ . The SLTs for  $F_1$  and  $F_2$  are shown in Figure 6.10. Matching of  $F_1 = x^2 + 3$  and  $F_2 = 3 + x^2$  starts with a call to `exact`, which uses the table in Figure 6.11 to match the three sub-expressions of  $F_1$  and  $F_2$  at any position on the typographical baseline of the two expressions. Each cell in Figure 6.11 corresponds to another call to `exact` on the cell pair. For the `LayoutLinearNode` non-terminals to match between  $F_1$  and  $F_2$ , exactly three pairs on the table in Figure 6.11 (i.e., as many as there are children in the matching `LayoutLinearNode` non-terminals) must match, which is indeed the case.

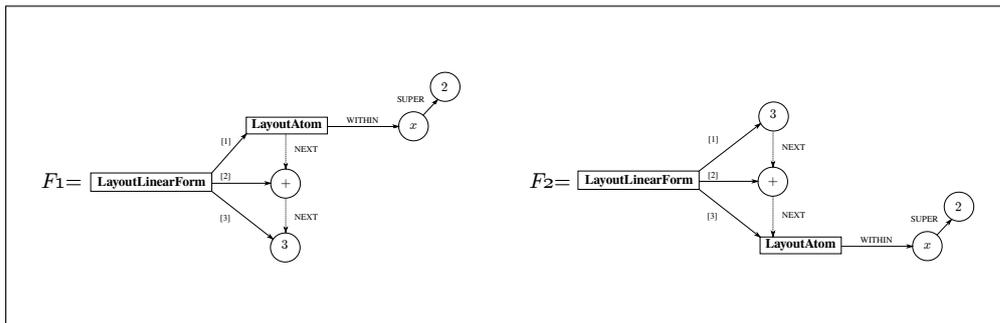


Figure 6.10: Equivalent SLTs under implicit commutativity for the formulae  $F_1 = x^2 + 3$  and  $F_2 = 3 + x^2$ .

| Step                                       |  | Direction | Result |
|--|--|-----------|--------|
| exact (LayoutLinearForm, LayoutLinearForm) |  | ROOT      | True   |

|                               |       | LayoutLinearNode <sub>2</sub> |       |       |
|-------------------------------|-------|-------------------------------|-------|-------|
|                               |       | $x^2$                         | +     | 3     |
| LayoutLinearNode <sub>2</sub> | 3     | False                         | False | True  |
|                               | +     | False                         | True  | False |
|                               | $x^2$ | True                          | False | False |

Figure 6.11: Position-independent matching of two `LayoutLinearNode` non-terminals in `exact` for formulae  $F_1 = x^2 + 3$  and  $F_2 = 3 + x^2$ .

The implicit commutativity assumption makes tree matching (Section 6.4.1) more flexible: it allows  $\alpha$ -equivalent expressions such as  $x^2 + 3$  and  $3 + y^2$  to be matched regardless of the order of

their sub-expressions. Such matches can be beneficial to both precision and recall, by increasing true positives and decreasing false negatives simultaneously. However, when applied to operators that are not commutative, the assumption can introduce false positive formula matches, affecting precision. I use implicit commutativity because I believe that overall the benefits should outweigh the cost, as most commonly used operators such as addition and multiplication are commutative, with only one notable exception, namely matrix multiplication.

### 6.1.3 SLT Matching with $\alpha$ -equivalence

The next-most complicated matching procedure decides whether two formulae are syntactically equivalent, by enforcing  $\alpha$ -equivalence (discussed on page 27). Recall that  $\alpha$ -equivalent formulae have the same structure but are expressed using a different vocabulary of symbols that is interchangeable (i.e., one formula can be converted to the other by substituting its symbols).

Realising  $\alpha$ -equivalence on SLTs is useful because it allows typed retrieval models to recognise semantically equivalent formulae expressed using different symbol conventions (i.e., different syntax).

To do so, I introduce the `alpha` operation that is realised using the same procedure as `exact` with some modifications. `alpha` uses two support operators on SLTs, `skolem` and `subs` to perform the  $\alpha$ -equivalence check. The operator `skolem` takes in an SLT as input and returns a *substitution map*. In general, a substitution map is a set of mappings:

$$\{D_1 \rightarrow R_1, \dots, D_n \rightarrow R_n\}, \quad (3)$$

where  $D_i \rightarrow R_i$  indicates that variable or identifier  $D_i$  is mapped to identifier or variable  $R_i$ . I will refer to the set of left-hand-side identifiers of a substitution map,  $\{D_1, \dots, D_n\}$ , as the *domain* and to the set of right-hand-side identifiers,  $\{R_1, \dots, R_n\}$ , as the *range*.

The `skolem` operator traverses the input SLT in a depth-first manner recording a substitution for each new mathematical identifier (domain) it encounters to a fresh number (range). Fresh numbers are generated for identifiers in the order they are encountered, or no new substitutions are recorded for identifiers that are already present in the substitution map.

Figure 6.12 illustrates how `skolem` will transform the SLT of  $x^2 + y$  (left) into a skolemisation substitution set (right). Figure 6.13 shows the steps `skolem` will take for  $x^2 + y$  to generate its skolemisation substitution map. For every new mathematical identifier in  $x^2 + y$  `skolem` encounters it records a mapping to a fresh number, producing the skolemisation substitution map  $\{\boxed{1} \rightarrow x, \boxed{2} \rightarrow +, \boxed{3} \rightarrow y\}$ .

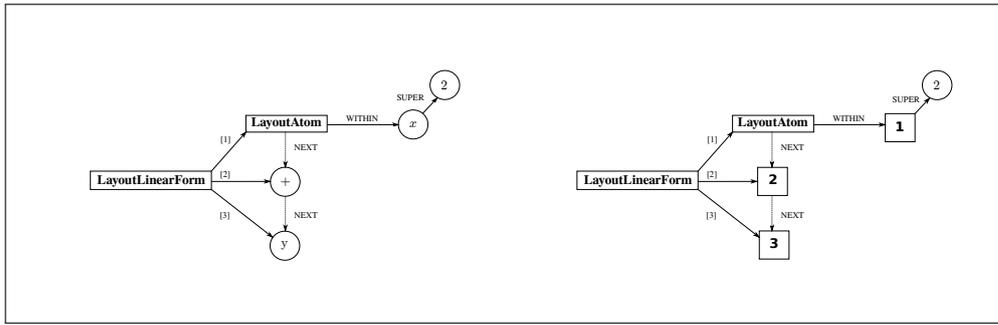


Figure 6.12: Left: the SLT for  $F = x^2 + y$ , right: skolemisation of  $F$ .

| Step  | Direction            | Result  |
|---|----------------------|---|
| <code>skolem(LayoutLinearForm, LayoutLinearForm)</code> | ROOT                 |   |
| <code>skolem(LayoutAtom)</code>                         | First child of ROOT  | $\{\}$  |
| <code>skolem(x)</code>                                  | WITHIN of LayoutAtom | $\{\boxed{1} \rightarrow x\}$   |
| <code>skolem(+)</code>                                  | Second child of ROOT | $\{\boxed{1} \rightarrow x, \boxed{2} \rightarrow +\}$                          |
| <code>skolem(y)</code>                                  | Third child of ROOT  | $\{\boxed{1} \rightarrow x, \boxed{2} \rightarrow +, \boxed{3} \rightarrow y\}$ |

Figure 6.13: Steps taken by *skolem* to produce the skolemisation map for  $x^2 + y$ .

The support operator `subs` takes two SLTs,  $F_1$  and  $F_2$  as input and returns a *conflict-free substitution map* that maps the variables or identifiers of  $F_1$  to those of  $F_2$ . A substitution map,  $S \equiv \{D_1 \rightarrow R_1, \dots, D_n \rightarrow R_n\}$ , is conflict-free if:

1. Each identifier  $D_i$  in its range is mapped onto at most one identifier in its domain, and
2. each identifier  $R_i$  in its domain has at most one unique identifier in its range mapped onto it.

I will now illustrate the concepts above using an example with `subs`. The SLTs for the  $\alpha$ -equivalent formulae  $F_1 = x^2 + y$  and  $F_2 = y^2 + x$  are shown in Figure 6.14. Figure 6.15 shows the steps that `subs` will take to map the identifiers in  $F_1$  to those in  $F_2$ .

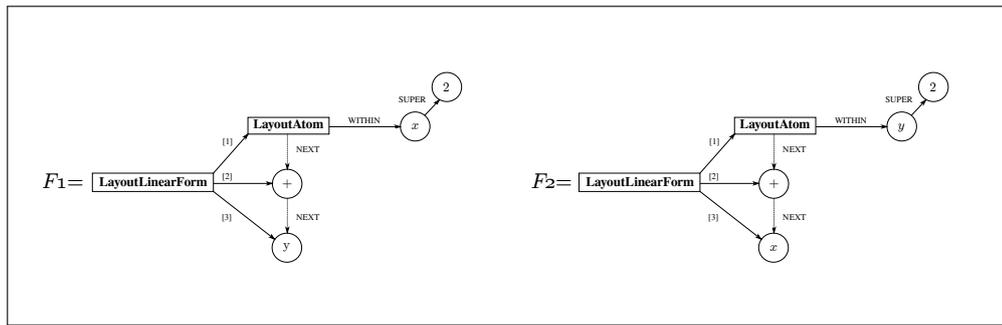


Figure 6.14: SLTs for the  $\alpha$ -equivalent formulae  $F_1 = x^2 + y$  and  $F_2 = y^2 + x$ .

| Step | Description                                    | Result  |
|------|--|---|
| 1    | $\text{skolem}(F_1)$ ,<br>$\text{skolem}(F_2)$ | Obtain skolemisation<br>substitution sets for $F_1, F_2$          |
| 2    | $\text{resolve}(S_1, S_2)$                     | Match the skolems in the<br>range to identifiers in the<br>domain |
| 3    | $\text{check}(R)$                              | Check the resulting<br>substitutions for consistency              |
| 4    | $\text{return}(R)$                             | return the map  |

Figure 6.15: Steps taken by *subs* to produce the mapping set for  $F_1 = x^2 + y$  and  $F_2 = y^2 + x$ .

First, *subs* produces the skolemisation maps for  $F_1$  and  $F_2$  using *skolem*. Next, it pairs substitutions from the two sets with the same skolem in their range and resolves their domain by substituting the range identifier with the variable in its domain, as shown in step 2 of Figure 6.14. Then, it checks the produced map for consistency and decides that the resulting map is consistent because the identifiers of  $F_1$  are mapped to exactly one identifier of  $F_2$  and vice-versa.

Having defined the support operators *skolem* and *subs* I will now describe the operation *alpha*, which returns *True* if two input SLTs,  $F_1$  and  $F_2$ , are  $\alpha$ -equivalent and *False* otherwise.

The *alpha* operation proceeds in the same way as *exact* with two modifications:

1. each recursive call to *alpha* also calls *subs* to its arguments to obtain the substitution map for the input SLTs.
2. in the case that both input SLTs are *LayoutLinearNode* non-terminals, position-independent matching will tabulate their children (as is the case for *exact*) but will only return *True* if the union of substitution maps in the matching cells is a consistent mapping.

To illustrate the modifications described above, consider again the  $\alpha$ -equivalent expressions  $F_1 = x^2 + y$  and  $F_2 = y^2 + x$ , the SLTs of which are shown in Figure 6.16.

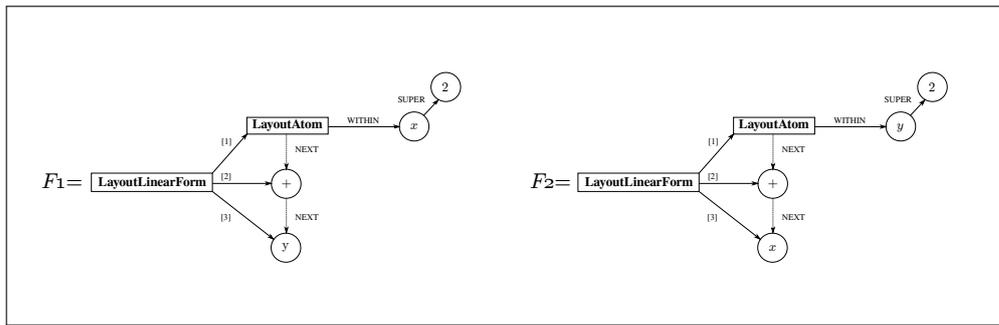


Figure 6.16: SLTs for the  $\alpha$ -equivalent formulae  $F_1 = x^2 + y$  and  $F_2 = y^2 + x$ .

The first call to `alpha` will pass the `LayoutLinearForm` of the first and the second SLT as its arguments as shown in step 1 of Figure 6.17. In step 2 a substitution map  $S = \{x \rightarrow y, + \rightarrow +, y \rightarrow x\}$  is produced by `subs` which is passed on to the recursive call to `alpha` in step 3. At this point, `alpha` encounters two sub-trees with `LayoutLinearNode` non-terminals, so position-independent matching is triggered.

| Step | Description  | Result  |
|------|--|---|
| 1    | <code>alpha (F<sub>1</sub>, F<sub>2</sub>, {})</code>      | Initial call  |
| 2    | <code>subs (F<sub>1</sub>, F<sub>2</sub>)</code>           | Obtain substitution map for $S = \{x \rightarrow y, + \rightarrow +, y \rightarrow x\}$ |
| 3    | <code>alpha (LayoutLinearNode, LayoutLinearNode, S)</code> | Recursive call  |
| 4    | Position-independent matching                              | True,<br>$\{x \rightarrow y, + \rightarrow +, y \rightarrow x\}$                        |
| 5    | <code>return (True, S)</code>                              | True,<br>$\{x \rightarrow y, + \rightarrow +, y \rightarrow x\}$                        |

Figure 6.17: Steps taken by `alpha` to perform  $\alpha$ -equivalent matching of  $F_1 = x^2 + y$  and  $F_2 = y^2 + x$ .

This creates the tabulation shown in Figure 6.18, where three sub-expressions are matched recursively, as expected. But this is not enough to verify the  $\alpha$ -equivalence of the expressions; the substitution maps will also have to be checked for conflicts. The substitution map produced from the table in Figure 6.18 is  $\{x \rightarrow y, + \rightarrow +, y \rightarrow x\}$ , which is conflict-free because the identifiers of  $F_1$  are mapped to exactly one identifier of  $F_2$  and vice-versa.

Now consider the two syntactically non-equivalent formulae  $F_1 = x^2 + y$  and  $F_2 = y^2 + y$ , whose SLTs are shown in Figure 6.19.

|                               |       | LayoutLinearNode <sub>1</sub> |                             |                             |
|-------------------------------|-------|-------------------------------|-----------------------------|-----------------------------|
|                               |       | $x^2$                         | +                           | $y$                         |
| LayoutLinearNode <sub>2</sub> | $y^2$ | True, $\{x \rightarrow y\}$   | False                       | False                       |
|                               | +     | False                         | True, $\{+ \rightarrow +\}$ | False                       |
|                               | $x$   | False                         | False                       | True, $\{y \rightarrow x\}$ |

Figure 6.18: Position-independent matching of two LayoutLinearNode non-terminals in *alpha* for formulae  $F_1 = x^2 + y$  and  $F_2 = y^2 + x$ .

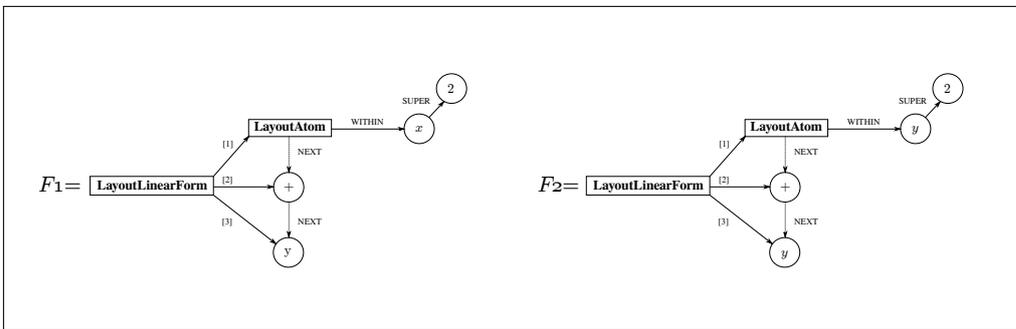


Figure 6.19: SLTs for the syntactically non-equivalent formulae  $F_1 = x^2 + y$  and  $F_2 = y^2 + y$ .

The *alpha* operation will perform the steps shown in Figure 6.20 for  $F_1$  and  $F_2$ . By step 2, *alpha* determines that the expressions are not  $\alpha$ -equivalent because their substitution map,  $S$ , has an inconsistency: there are two variables from  $F_1$ ,  $x, y$  mapped onto variable  $y$  in  $F_2$ .

The matching process is allowed to continue to the position-independent matching step. At this step, shown in Figure 6.21, the conflict is isolated and the process terminates without success.

| Step | Description   | Result  |
|------|---|---|
| 1    | <code>alpha(F<sub>1</sub>, F<sub>2</sub>, {})</code>      | Initial call  |
| 2    | <code>subs(S<sub>1</sub>, S<sub>2</sub>)</code>           | Obtain substitution map for $F_1, F_2$<br><br>$S = \{x \rightarrow y, + \rightarrow +, y \rightarrow y\}$ |
| 3    | <code>alpha(LayoutLinearNode, LayoutLinearNode, S)</code> | Recursive call  |
| 4    | Position-independent matching                             | False,<br>$\{x \rightarrow y, + \rightarrow +, y \rightarrow y\}$   |
| 5    | <code>return(False, S)</code>                             | False,<br>$\{x \rightarrow y, + \rightarrow +, y \rightarrow y\}$   |

Figure 6.20: Steps taken by *alpha* to perform  $\alpha$ -equivalent matching of  $F_1 = x^2 + y$  and  $F_2 = y^2 + y$ .

|                               |       | LayoutLinearNode <sub>1</sub> |                             |  |
|-------------------------------|-------|-------------------------------|-----------------------------|--|
|                               |       | $x^2$                         | $+$                         | $y$                                    |
| LayoutLinearNode <sub>2</sub> | $y^2$ | True, $\{x \rightarrow y\}$   | False                       | False                                  |
|                               | $+$   | False                         | True, $\{+ \rightarrow +\}$ | False                                  |
|                               | $y$   | False                         | False                       | False, conflict: $\{y \rightarrow y\}$ |

Figure 6.21: Position-independent matching of two `LayoutLinearNode` non-terminals in *alpha* for formulae  $F_1 = x^2 + y$  and  $F_2 = y^2 + y$ .

This is not the first time we have seen that my SLT extensions described in Section 6.1.1 have a positive effect. I will here exemplify in even more detail how the implementation of the `exact` and `alpha` operations produce this effect. The first reason is that `exact` is likely to reject two distinct SLTs earlier if they are composed of different structural idioms at some point in their sub-trees. It does so by recognising that the corresponding non-terminals are not the same.

To illustrate, suppose that we need to check whether

$$\frac{x^2}{\sqrt{z}} \tag{4}$$

is an exact match to the expression

$$\frac{x^2 + y}{\sqrt{z}} \quad (5)$$

For formula 4, Tangent will produce the SLT shown in the left part of Figure 6.22, while my SLT representation is the one shown in the right of Figure 6.22. Formula 5 will be represented by Tangent using the SLT shown in the left part of Figure 6.23, while my SLT representation is the one shown in the right of Figure 6.23.

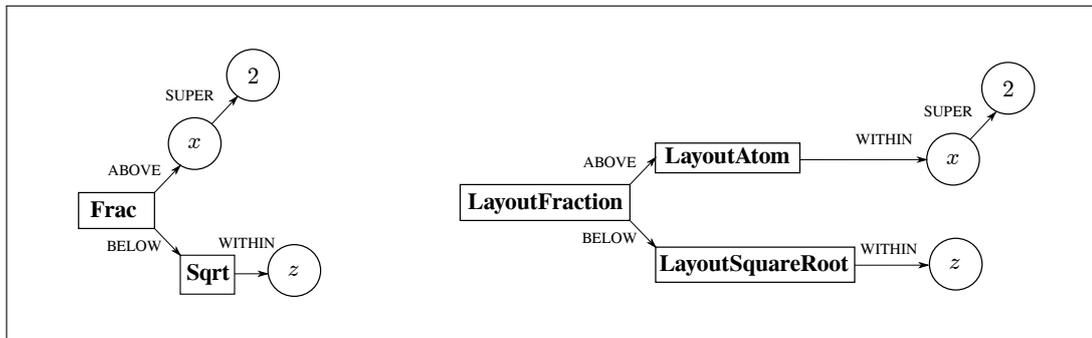


Figure 6.22: SLTs for  $\frac{x^2}{\sqrt{z}}$ . Left: Tangent's, right: mine.

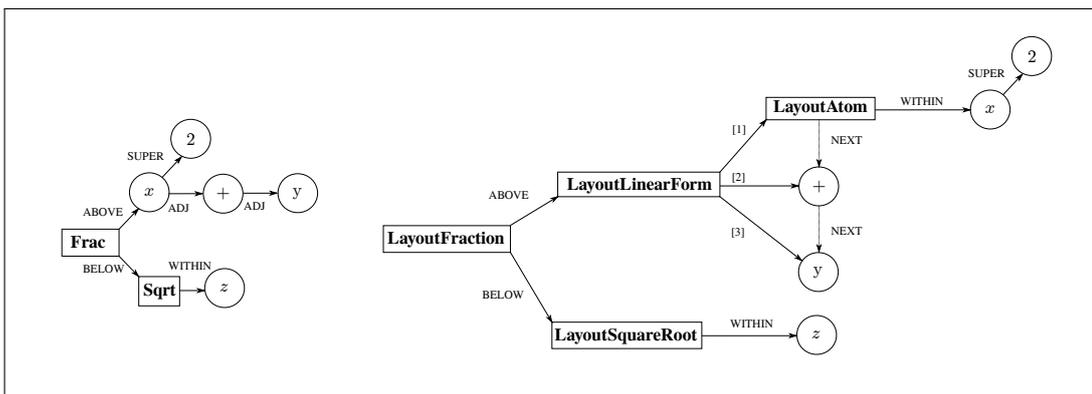


Figure 6.23: SLT for  $\frac{x^2+y}{\sqrt{z}}$ . Left: Tangent's, right: mine.

The operation `exact` implemented using the SLTs in the left part of Figures 6.22 and 6.23 will first simultaneously match the **BELOW** branches with success. Then, when looking at the **SUPER** branches, it will match the  $x$  and  $2$  nodes before recognising that one SLT has an **ADJ** branch, while the other does not. This will lead to a rejection of the match.

In contrast, the operation `exact` implemented using my SLTs, shown in the right part of Figures 6.22 and 6.23, will recognise earlier that the SLTs are not exact matches without having to match specific non-terminal nodes (such as  $x$  or  $2$ ) because the **SUPER** branches of the two SLTs have different non-terminal nodes.

The second reason is that the introduction of the non-terminal `LayoutLinearNode` groups together those sub-expressions that are laid out on the same typographical baseline. This allows

exact to implicitly know which sub-expression pairs need to be tabulated for matching under the implicit commutativity assumption. In contrast, without my SLT extensions, an additional step would be required on the SLTs to determine which symbols and non-terminals are on the same baseline following ADJ edges between nodes.

### 6.1.4 Recursive SLT Matching

It is useful to be able to find instances of one expression occurring as sub-expressions of another, larger expression, an operation I call recursive SLT matching.

Given two SLTs,  $s$  and  $t$ , `recursive` finds all instances of  $s$  in  $t$  by first identifying all nodes in  $t$  that are of the same kind as the root node of  $s$ . Then, each of these nodes is matched against  $s$ , using either `exact` or `alpha`.

To illustrate, suppose we want to find all instances of  $F_1 = x^2$  in  $F_2 = \frac{x^2}{\sqrt{z}}$ . The SLTs for  $F_1$  and  $F_2$  are shown in Figure 6.24.

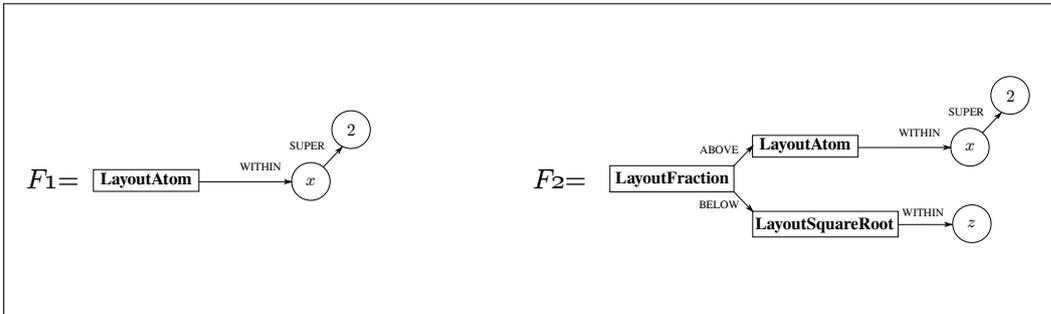


Figure 6.24: Left: SLT for  $x^2$ , right: SLT for  $\frac{x^2}{\sqrt{z}}$ .

The operation `recursive( $F_1, F_2$ )` will identify the ABOVE branch of the SLT for  $F_2$  (Figure 6.24, right) as a sub-expression that is a potential match to  $F_1$  because their sub-trees both have `LayoutAtom` nodes as their roots. Then, it will call `exact` (or `alpha`) and if the result of the call is `True`, the matching node in  $F_2$  will be recorded as a match.

This example highlights another advantage of my SLT extensions, namely that they make implementing recursive matching easier: finding all instances of  $x^2$  in  $\frac{x^2}{\sqrt{z}}$  now only involves matching the former against the `LayoutAtom` idioms of the latter. Without my extensions, recursive matching would involve checking all possible sub-trees of  $\frac{x^2}{\sqrt{z}}$ .

## 6.2 Unification over SLTs

The next big idea in this thesis is type unification. Type unification models semantic similarity between two  $\alpha$ -equivalent SLTs. It is a semantic step that operationally runs on top of the `exact`, `alpha` and `recursive` operations from Section 6.1.

Specifically, in addition to the  $\alpha$ -equivalence check performed by `alpha`, type unification involves examining syntactically matching sub-expressions in two SLTs (i.e., matching typed

terminal or non-terminal nodes), to determine whether their types can be unified using the `unify` operator below, which implements the `type-unif` inference rule from Chapter 4 (page 81).

Given two types  $T_1$  and  $T_2$ , the unifying type  $T$  is a type in my type dictionary such that  $T$  is either (a) a common supertype of  $T_1$  and  $T_2$  or (b) is a member of the intersection of similar types to  $T_1$  and  $T_2$  in the type embedding space:

$$T \in embsim(T_1) \cap embsim(T_2)$$

If  $|embsim(T_1) \cap embsim(T_2)| > 1$ , then  $T$  is the type that maximises the mean cosine similarity of  $T$  to  $T_1$  and  $T$  to  $T_2$  in the embedding space:

$$\frac{sim(T, T_1) + sim(T, T_2)}{2}$$

I define a `unify` operator that returns `True` if a unifying type between  $T_1$  and  $T_2$  can be found as follows:

$$unify(T_1, T_2) = \begin{cases} \text{True,} & \text{if } |embsim(T_1) \cap embsim(T_2)| > 0 \vee h(T_1, T_2) > 0 \\ \text{False,} & \text{otherwise,} \end{cases}$$

where  $embsim(T_1)$  and  $embsim(T_2)$  return the most similar types to  $T_1$  and  $T_2$ , respectively; and  $h(T_1, T_2)$  returns the set of common hypernyms to  $T_1$  and  $T_2$  on the KTST (Known Types Suffix Trie, cf. section 4.2.3).

In the context of typed retrieval, type unification is a form of query expansion: it enables matching of STL nodes that would otherwise be considered different. To illustrate this, I return to the example from the beginning of this chapter. Consider again the expressions  $F_1 = hom(G_1, G_2)$  and  $F_2 = home(R_1, R_2)$ , with  $G_1, G_2 :: GROUP$  and  $R_1, R_2 :: RING$ , represented by the SLTs shown in Figure 6.25.

| Formula         | Untyped SLT | Typed SLT |
|-----------------|-------------|-----------|
| $hom(G_1, G_2)$ |             |           |
| $hom(R_1, R_2)$ |             |           |

Figure 6.25: Untyped and typed SLTs for the formulae  $hom(G_1, G_2)$  and  $hom(R_1, R_2)$  (repeated from page 129).

| Step |  | Result  |
|------|--|---|
| 1    | Initial call to <code>alpha (LayoutLinearForm, LayoutLinearForm)</code>  | <code>True</code> , $\{G_1 \rightarrow R_1, G_2 \rightarrow R_2\}$                                |
| 2    | Call to <code>subs</code>  | $\{G_1 \rightarrow R_1, G_2 \rightarrow R_2\}$  |
| 3    | Tabulate sub-expressions for matching under implicit unification   | Sub-expressions matched under substitution<br>$\{G_1 \rightarrow R_1, G_2 \rightarrow R_2\}$      |
| 4    | With $G_1, G_2 :: GROUP$ and $R_1, R_2 :: RING$ , <code>unify(GROUP, RING)</code> twice  | Both calls to <code>unify</code> return <code>True</code> , since $RING \in emb\text{sim}(GROUP)$ |
| 5    | Initial call to <code>alpha</code> returns <code>True</code> since <code>alpha</code> equivalence is verified in step 3 and type unification succeeded in step 4 | <code>True</code> , $\{G_1 \rightarrow R_1, G_2 \rightarrow R_2\}$                                |

Figure 6.26: Steps to match  $hom(G_1, G_2)$  and  $hom(R_1, R_2)$  with  $\alpha$ -equivalence checks and type unification.

Figure 6.26 shows the steps that the `alpha` operator will take to determine that  $hom(G_1, G_2)$  and  $hom(R_1, R_2)$  are  $\alpha$ -equivalent and can be type-unified. The SLTs of both expressions have instances of `LayoutLinearForm` non-terminals as their roots, which are passed as arguments to `alpha` in the initial call.

Then, in steps 2 and 3, `alpha` tabulates the child SLTs of the non-terminals and recursively calls itself for each cell in the tabulation. This tabulation is shown in Figure 6.27. Steps 2 and 3 return `True` since five  $\alpha$ -equivalent pairs have been discovered with consistent substitutions for the four identifiers  $G_1, G_2, R_1, R_2$  (cf. Table 6.27).

|                               |            | LayoutLinearForm <sub>1</sub> |       |                                       |       |                                       |       |
|-------------------------------|------------|-------------------------------|-------|---------------------------------------|-------|---------------------------------------|-------|
|                               |            | <i>hom</i>                    | (     | $G_1$                                 | ,     | $G_2$                                 | )     |
| LayoutLinearForm <sub>2</sub> | <i>hom</i> | True                          | False | False                                 | False | False                                 | False |
|                               | (          | False                         | True  | False                                 | False | False                                 | False |
|                               | $R_1$      | False                         | False | True, $\{R_1 \rightarrow G_1\}$       | False | False since $\{G_2 \rightarrow R_2\}$ | False |
|                               | ,          | False                         | False | False                                 | True  | False                                 | False |
|                               | $R_2$      | False                         | False | False since $\{R_1 \rightarrow G_1\}$ | False | True, $\{R_1 \rightarrow R_2\}$       | False |
|                               | )          | False                         | False | False                                 | False | False                                 | True  |

Figure 6.27: Position-independent matching of  $hom(G_1, G_2)$  and  $hom(R_1, R_2)$  with  $\alpha$ -equivalence checks.

The fourth step in Table 6.26 takes the substitutions discovered in the previous steps,  $\{G_1 \rightarrow$

$R_1, G_2 \rightarrow R_2\}$  and makes use of the type assignments  $G_1, G_2 :: GROUP$  and  $R_1, R_2 :: RING$  to perform type unification as follows:

1. Knowing  $\{G_1 \rightarrow R_1, G_2 \rightarrow R_2\}$  and  $G_1, G_2 :: GROUP, R_1, R_2 :: RING$ , we deduce that  $\{GROUP \rightarrow RING, GROUP \rightarrow RING\}$ .
2. We need to show that  $\text{unify}(GROUP, RING) \wedge \text{unify}(GROUP, RING) \equiv \text{True}$ .
3.  $\text{unify}(GROUP, RING) \wedge \text{unify}(GROUP, RING)$  reduces to  $\text{True} \wedge \text{True}$  because  $RING \in \text{embsim}(GROUP)$ .

As a result, type unification allows the syntactically equivalent expressions  $\text{hom}(G_1, G_2)$  and  $\text{hom}(R_1, R_2)$  to be unified because the variables  $G_1, G_2$  and  $R_1, R_2$  are (a) in isomorphic positions in the two formulae and (b) their types,  $GROUP$  and  $RING$ , are similar in the type embedding space.

Now we are in a position to turn to the practical aspect of how to build an MIR system based on the matching operations just introduced.

### 6.3 Practical Typed MIR

To realise typed retrieval in the real world we need two additional practical procedures. The first is type disambiguation, a process for deciding the type that will be used to denote each unique variable in the discourse (document or query). To remind the reader, the variable typing process described in Chapter 5 produces typings. Typings are a pairing of two entities:

- a *variable instance*, i.e., an instance of a variable encountered in the textual context, whereby each variable instance is represented by a distinct SLT. However, as all variable instances of the same conceptual variable (or *unique variable*) are identical, so are their SLTs; and
- a textual description of a *type* encountered in the discourse.

For typed retrieval to be realised it is necessary to group typings by their unique variable, look at the set of extracted types for that variable and assign one type across all of its instances (SLTs). This is the goal of type disambiguation procedure. Type disambiguation produces a mapping that maps a set of equivalent SLTs (that represent a unique variable) to one type.

The second procedure is the typing of SLTs. Note that all SLTs representing variables have already been typed during type disambiguation, so the only SLTs left to be typed are the non-variable SLTs (corresponding to all other formulae, including equalities and other mathematical statements). This step uses the output of type disambiguation to produce typed SLTs by replacing instances of each variable in every formula with its type. The type disambiguation and typed SLT creation procedures are applied to both documents and queries. Both procedures are described in more detail in the two sections that follow.

In Section 6.3.3 I address two additional practical difficulties with realising tree matching retrieval models. The first difficulty is that scoring document–query formula pairs using tree matching at query time is infeasible because tree matching is an expensive operation. To address this difficulty I made a compromise: instead of indexing documents and scoring them against a query at query time, I opted to compute a set of features for each document–query formula pair in my collection, in an offline manner.

Features include direct outcomes to the SLT operations, such as `exact` and `alpha`, but also metrics, such as number of nodes matched in a pair or number of type-unifiable nodes, that allow me to compute heuristics (e.g., proportion of nodes matched or proportion of unifiable variables) for scoring. The features are computed once for every pair in my test collection using my tree matching algorithms and are stored offline in a table. Using the information in the table, I can combine the stored features to form arbitrary tree matching scoring methods, allowing me to experiment with tree matching models offline.

The second difficulty is that tree matching using the operators from Section 6.1 results in a binary outcome, but MIR requires continuous scores that model the degree of similarity, to enable more flexible scoring. This difficulty is addressed by the design of my tree matching scoring functions, which I discuss in Section 6.3.3. My scoring functions combine scores coming from five aspects of tree-based similarity into a single score for the symbolic context.

I will discuss the features, the aspects modelled by my features and the design of my tree matching scoring functions in Section 6.3.3.

### 6.3.1 Type Disambiguation

Disambiguation becomes necessary in typed retrieval because (a) in real texts, a variable can have several types in one larger discourse, here a document or a query (b) variable typing as described in Chapter 5 extracts denotations assigned in independent sentences, i.e. it types individual instances of variables, and (c) for practical Math IR, I want to enforce an ideal situation where every symbol has at most one type in a document.

In this work, I model the link between modalities at the document level, as is common in document-level retrieval. But there are many ways the document–query score can be realised, a decision that also depends on the scoring objectives of the MIR system. For example, an MIR system designed to retrieve theorems in mathematical papers could model the link between modalities at the sub-document level by grouping and typing variables at the context of sections (e.g., section-by-section) or mathematical blocks (Section 2.5).

I will first describe why I want to enforce point (c) above. The assumption of “one variable, one type” reduces storage and computational demands as well as the complexity of the design of typed retrieval models. Without this assumption, the possibility space of type unification (introduced in Section 6.2), which is based on pair-wise comparison of types between the variables of  $\alpha$ -equivalent expressions, would become prohibitively large.

Let us now look at point (b), and why variables can have more than one type in a document, a

phenomenon called *overloading*. There are two cases when symbols can get overloaded:

1. The mathematical discourse can have many localised contexts in the form of blocks where new assumptions about objects may be introduced (Ganesalingam, 2008). Mathematicians introduce a symbol for a mathematical object for the purpose of mathematical reasoning. A second object may appear during the reasoning which is related to the original object in some form (e.g. an abstraction or a specialisation), but the mathematician now uses the same symbol to refer to the second object too, i.e., she overloads the symbol.
2. Symbols may be reused across loosely related sections of a mathematical document to make content easier to follow. For example, in Section 1 of Lorenzin (2020) the symbol  $\mathcal{A}$  is declared to be a heart:

**Lemma 1.5** *Let  $\mathcal{A}$  be a heart (of a bounded  $t$ -structure) on a triangulated category  $\mathcal{T}$  with  $\dim_{\mathcal{T}} \mathcal{A} \leq 1$ , meaning that  $\text{Hom}_{\mathcal{T}}(\mathcal{A}, \mathcal{A}[n]) = 0$  for any  $n \geq 2$ . Then  $\mathcal{T}$  is uniquely determined by  $\mathcal{A}$  up to exact equivalence.*

Then, in Section 2,  $\mathcal{A}$  is re-declared as an abelian category:

**Definition 2.1.** Let  $\mathcal{A}$  be an abelian category. The elements of the group  $\text{Ext}^n(A, B) \cong \text{Hom}_{\mathcal{D}^b(\mathcal{A})}(A, B[n])$  are  $n$ -extensions for  $n > 0$ ...

Overloading thus results in a variable having more than one type assignments throughout the discourse (query or document). My approach aims to make balanced decisions when deciding which type to assign to a variable: these types are hopefully close to the types of the text, and they should not be too specialised. The type disambiguation algorithm I developed is based on three constraints:

1. the type  $t$  assigned to a variable  $v$  should either be one that is extracted from the text or one of its supertypes. I introduced this constraint to my algorithm because I want the meaning of variable  $v$  to not deviate too much from that intended in the source text.
2. the type  $t$  assigned to a variable  $v$  from its set of typings should neither be too specialised nor too general. This constraint gives room for type unification (Section 6.2) to unify variables in two expressions by either generalising or specialising their types. Types that are too specialised might have many supertypes on the known type suffix trie (KTST) but few similar types in the embedding space while types that are too abstract might have no supertypes but many similar types in the embedding space.
3. The frequency by which  $v$  is assigned  $t$  in the context (an MREC document or query) is a good indicator whether  $v$  is of type  $t$ . Types that are too specialised are those that are located deep down the KTST. Such types should be rarely observed in a corpus. In other words, I use the frequency of assignment between  $t$  and  $v$  in the source context as positive bias.

I will now express the type disambiguation process in a more formal way. A document  $D$  is a collection of instances of formulae  $E$ , some of which are variables. The variable instances are represented as SLTs<sup>6</sup> and make up the set  $S = \{s_1, \dots, s_n\}$ . The output of variable typing gives us on top of this a set of variable typings, one for each variable instance  $\{(s_1, t_1), \dots, (s_m, t_m)\}$ . Variable typing returns positive edges for every instance of a header variable (cf. Section 5.4) in  $D$ . As a result, there may be many pairs  $(s_i, t_i)$  associated with a particular header variable  $v \in V$ .

We desire a set of unique variables  $V = \{v_1, \dots, v_m\}$  which is not known a priori and must be computed from the set of variable instances  $S$ . The process of type disambiguation computes  $V$  by approximating it using  $\hat{V}$ , a partitioning of  $S$ . Any SLT in a partition can now be used to represent the unique variable that the partition models. For every  $s \in \hat{V}$  it enumerates all candidate types and makes decisions about which type to assign to the partition as a whole. Having decided a type for each equivalence class, every SLT in the class is labelled with that type. Finally, for every non-variable formula in the document, SLT typing will recursively match every sub-variable to an equivalence class and label the sub-variable with its type.

In detail, the process is as follows:

**Step 1** Approximate  $V$  by  $\hat{V}$ , where  $\hat{V}$  is a set of equivalence classes  $C_v$ . This is done by grouping typings  $\{(s_1, t_1), \dots, (s_m, t_m)\}$  by equality of SLTs  $s_i =_{SLT} s_j$  (realised by `exact`, Section 6.1.2).

An equivalence class  $C_v$  is a tuple

$$C_v = (\{s_1, \dots, s_k\}, [t_1, \dots, t_k]), \text{ such that } \forall_{i,j} s_i \in \{s_1, \dots, s_k\}, s_i =_{SLT} s_j.$$

where  $[t_1, \dots, t_k] \equiv T$  is the bag of candidate types for  $C_v$ ; the same type might appear more than once in typings belonging to  $C_v$ .

**Step 2** Compute frequency maps for classes in  $\hat{V}$ . For each class  $C_v \in \hat{V}$ , compute the frequency by which every unique type occurs in  $C_v$ . Conceptually,  $C_v$  is now transformed into:

$$C_v = (\{s_1, \dots, s_k\}, \{t_1 \rightarrow freq(t_1), \dots, t_p \rightarrow freq(t_p)\}), \\ \text{such that } \forall_{i,j} s_i \in \{s_1, \dots, s_k\}. s_i =_{SLT} s_j.$$

The term  $freq(t)$  refers to the number of occurrences of  $t$  in  $T$ . The bag of types of  $C_v$  now becomes a type frequency map.

**Step 3** Type Disambiguation step. The type disambiguation algorithm,  $\hat{\delta}$  (equation 6) returns the type,  $T_i$ , that is to be the sole type of SLTs in  $C_i$ , as follows:

---

<sup>6</sup>Technically, I create an SLT for each variable instance in the preprocessing stage of variable typing, which translated from the source MathML.

$$\hat{\delta} = \operatorname{argmax}_{t \in T} [0.9 \times \operatorname{freq}(t, v) + 0.1 \times h(t)] \quad (6)$$

where  $h(t)$  is the number of supertypes of  $t$  observed on the KTST.

Every SLT in  $C_i$  is subsequently annotated as having the type  $T_i$ . After type disambiguation,  $C_i$  is reduced to a simpler tuple

$$C_i = (\{s_1, \dots, s_k\}, T_i).$$

One might consider whether the type embedding space introduced in chapter 4.3 could have been used in some form to aid type disambiguation, but there is a danger that the embedding space might introduce related types that are not supertypes of  $t$ , violating constraint (1) above. I therefore chose not to make use of any such information in Equation 6.

### 6.3.2 Typing of non-variable SLTs

The next step in the procedure now concerns how all remaining untyped SLTs are converted into typed SLTs (cf example in Fig. 6.2 on page 129).

**Step 4** Produce typed SLTs for all remaining non-variable formulae in  $D$  by typing the constituents of non-variable formulae in  $D$ . The set of non-variable formulae is  $F = E \setminus \hat{V}$  and is defined as follows:

$$F = \{e_i \mid \forall_{e_i \in E, s \in \hat{V}} s \neq_{SLT} e_i\}$$

In order to type formulae  $f_i \in F$ , my algorithm matches each constituent of SLT  $f_i$  to one of the typed variables in  $\hat{V}$ . If a sub-expression of  $f_i$  is matched to an SLT in  $C_i$ , then that sub-expression is annotated with type  $T_i$ . This is a recursive process realised by the operation *recursive* described in Section 6.1.4.

For example, the typings  $a, b :: VECTOR$  for variables  $a$  and  $b$  are propagated to larger expressions such as  $a + b$ : the SLT for  $a + b$  can be annotated to encode  $a :: VECTOR + b :: VECTOR$ . This process of SLT construction produces information that is used in typed matching and scoring.

At the end of typed SLT creation, the following information is available:

1. A set of equivalence classes of type assignments, one for each unique variable

$$\hat{V} \equiv \{(\{SLT_{1,1}, \dots, SLT_{1,k}\}, T_1), \dots, (\{SLT_{i,j}, \dots, SLT_{i,p}\}, T_q)\}$$

2. The set of of all typed SLTs in  $D$  updated with type annotations,  $\{s'_1, \dots, s'_n\}$

This output can then be used to either generate a typed index (e.g., a typed Tangent index, cf. Section 6.4.3) or, when the procedure above is applied to both documents and queries, to score

the symbolic modality of document  $D$  with respect to query  $Q$ .

### 6.3.3 Scoring Tree Similarity

I apply my tree matching algorithms on the CUMTC once and record 27 tree matching features. These features are calculated for each document–query formula pair  $(QF_i, DF_j)$ , where  $QF_i$  and  $DF_j$  are typed SLTs produced by the procedure described in Section 6.3.2.

I organise my features into five groups: (a) structural similarity, (b) lexical similarity, (c)  $\alpha$ -equivalence, (d) semantic similarity and (e) recursive similarity. These classes and the features associated to them model the five different ways one formula can be related to another. The motivation behind this grouping of features will become clear later in this section. I also record general pair properties, which I classify under the general features group. In more detail, the feature groups are as follows:

- *General features* record information to identify the formulae in the query and document (e.g., unique identifiers) and general properties such as number of nodes (sNodes, tNodes), number of identifiers (idS, idT) and number of identifiers annotated with types (sTypable, tTypable).
- *Structural features* model structural similarity between  $QF_i$  and  $DF_j$ . The more terminal and non-terminal nodes shared at the same positions of the SLTs of  $QF_i$  and  $DF_j$ , the higher the structural similarity between the formulae becomes. Structural similarity features are extracted using variants of the `exact` operation (with implicit commutativity) that penalise differences in the two SLTs.
- *Lexical features* model lexical similarity using the `subs` operation. Lexical similarity rewards a pair of SLTs if they are (a) structurally similar and (b) the same symbols are encountered at the identical positions in the two SLTs.
- The  *$\alpha$ -equivalence indicator feature* is the main feature recording information about syntactical similarity. This feature expresses whether the two formulae are  $\alpha$ -equivalent. This feature is determined using the `alpha` operation.
- *Semantic features* model semantic similarity using semantic overlap (Section 6.4.2) and typed unification (as demonstrated in Section 6.2).
- *Embedded (or Recursive) features* model implicit similarity and contribute to the overall similarity between  $QF_i$  and  $DF_j$  in the case where one or more instances of one formula can be found as sub-expressions in the other. Embedded features are produced using the `recursive` matching operation.

The structural, lexical,  $\alpha$ -equivalence indicator and semantic feature groups model direct similarity between two SLTs (i.e., how similar two symbol layout trees as atomic tree structures

are). However, they do not reflect any recursive similarity attributable to one SLT being a constituent of the other (cf. Section 6.1.4).

Table 6.1 lists all direct comparison features. The values of the features in Table 6.1 are maximised when two SLTs have the same tree structure, are  $\alpha$ -equivalent with respect to their variables, and whose types are unifiable via type unification.

| Feature  | Description   |
|--|---|
| <b>General Features</b>                        |   |
| qid  | Query formula ID  |
| did  | Document formula ID   |
| sNodes   | Number of nodes in the query formula  |
| tNodes   | Number of nodes in the document formula   |
| sTypable                                       | Number of nodes with types in the query formula   |
| tTypable                                       | Number of nodes with types in the formula   |
| idS  | Number of nodes representing identifiers in the query formula   |
| idT  | Number of nodes representing identifiers in the document formula  |
| <b><math>\alpha</math>-equivalence Feature</b> |   |
| isMatch  | Indicator whether the query formula is $\alpha$ -equivalent to the document formula (1 = True, 0 = False) |
| <b>Structural Features</b>                     |   |
| h_total  | Total number of nodes compared  |
| h_match  | Total number of nodes that matched  |
| <b>Lexical Features</b>                        |   |
| h_idtotal                                      | Total number of identifiers compared  |
| h_idmatch                                      | Total number of identifiers that matched  |
| <b>Type Unification Features</b>               |   |
| h_typedtotal                                   | Total number of typed nodes compared  |
| h_typedmatch                                   | Total number of typed nodes that matched directly   |
| h_typedsuffix                                  | Total number of typed nodes matched through suffix trie traversal   |
| h_typedexp                                     | Total number of typed nodes matched through type expansion  |

Table 6.1: Features recorded for direct similarity between two SLTs.

The features listed in Table 6.2 model recursive similarity between two SLTs. The values of recursive features for the SLT pair  $(QF_i, DF_j)$  are maximised when one of the SLTs has one or more instances of the other as its constituents. For example, the more occurrences of  $QF_i$  found in  $DF_j$ , the more matches are recorded resulting in larger values for the feature `rec_matched`.

A tree matching scoring function,  $M(Q, D)$ , is responsible for computing a score that represents how similar the symbolic modality of document  $D$  is to the symbolic modality of query  $Q$ . Designing tree matching models using my features amounts to designing  $M(Q, D)$  functions that produce a score for the symbolic modalities of  $D$  and  $Q$  using features for all document–query formula pairs. The general form of a scoring function is shown in Figure 6.28, but other scoring

| Feature                   | Description   |
|---------------------------|---|
| <b>Recursive Features</b> |   |
| nTotal                    | Number of sub-expressions in the larger formula compared to the smaller formula.  |
| nMatched                  | Number of instances of the smaller formula found in the larger formula  |
| rec_total                 | Cumulative total of nodes compared to find instances of the smaller expression in the larger                              |
| rec_matched               | Cumulative total number of nodes matched while looking for instances of the smaller expression in the larger              |
| rec_idtotal               | Cumulative total number of identifiers compared   |
| rec_idmatch               | Cumulative total number of identifiers matched  |
| rec_typed_total           | Cumulative total number of typed nodes compared   |
| rec_typed_matched         | Total number of typed nodes matched while looking for instances of the smaller expression in the larger                   |
| rec_typed_suffix          | Number of typed nodes matched due to suffix traversal while looking for instances of the smaller expression in the larger |
| rec_typed_exp             | Number of typed nodes matched due to type expansion while looking for instances of the smaller expression in the larger   |

Table 6.2: Features recorded for modelling recursive similarity between two SLTs attributed to instances of one being constituents of the other.

architectures can be constructed using my tree matching features.

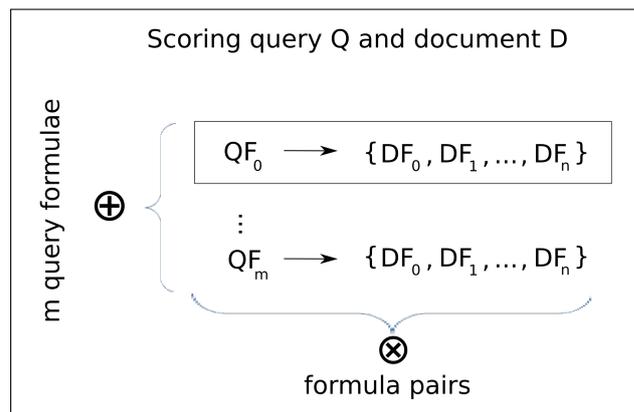


Figure 6.28: General form of a scoring function in typed retrieval.

In my tree matching scoring architecture there are five steps taken to score a document with respect to a particular query. The first step is to pair each query formula  $QF_i$  with all document formulae  $DF_j$  to form a map  $QF_i \rightarrow \{DF_0, \dots, DF_n\}$  for  $QF_i$ , as shown by the rectangle in Figure 6.28. The remaining steps correspond to decisions I can make when designing new models:

1. Decide on the pair similarity function,  $score(QF_i, DF_j)$ , that models the similarity of each individual document–query formula pair using the pair features.

The function  $score(QF_i, DF_j)$  is applied to each pair in the mapping  $QF_i \rightarrow \{DF_0, \dots, DF_n\}$  to produce a similarity score for every document formula with respect to  $QF_i$ .

2. Decide on the function  $\otimes$  (see Figure 6.28) that determines which document formula  $DF_j$  is most similar to query formula  $QF_i$  in its mapping of pairs. This function makes use of scores produced by  $score(QF_i, DF_j)$  to make its decision and returns a measure  $score(QF_i|D)$  that models how similar the document symbolic modality is to query formula  $QF_i$ ;
3. Decide on the function  $\oplus$  to aggregate the set of individual  $score(QF_i|D)$  scores produced by  $\otimes$  into the score symbolic context score for  $Q$  and  $D$ .
4. Apply any transformations to the scores produced by the previous step and return the result as the output of  $M(Q, D)$ .

I will now discuss the role of the pair similarity function,  $score(QF_i, DF_j)$ , in my tree matching models. I adopted the same general form for  $score(QF_i, DF_j)$  in all of my tree matching models (Section 6.4.1). My realisations of the function  $score(QF_i, DF_j)$  compute three sub-scores, one for each group for direct similarity features and three sub-scores for recursive features. I label the direct similarity sub-scores as

$$structural_{(QF_i, DF_j)}, lexical_{(QF_i, DF_j)} \text{ and } semantic_{(QF_i, DF_j)},$$

which model structural, lexical and semantic similarity between two SLTs, respectively. Three recursive sub-scores

$$structural_{R(QF_i, DF_j)}, lexical_{R(QF_i, DF_j)} \text{ and } semantic_{R(QF_i, DF_j)},$$

measure structural, lexical and semantic similarity (respectively) when one formula (e.g., from the query) is a constituent of another (e.g., from the document).

The motivation behind grouping features and breaking-down my pair similarity function into sub-scores is two-fold. First, having each kind of similarity contributing independently to the similarity score of a pair makes my models robust. For example, types for SLT constituents cannot always be extracted (as discussed in the beginning of this chapter and in Section 5.4) and this can result in low or zero scores for semantic similarity. However, since the other kinds of similarity contribute independently, the final score for a pair will be non-zero. This is desirable as we want to avoid producing zero scores for many documents, effectively putting them in the same bottom bin.

Second, it allows me to disable or alter the similarity from particular aspects when designing models, so that the aspects' effect can be measured in controlled experiments. For example, I can design an experiment that measures the effect of type unification (i.e., semantic aspect) by comparing two models: one that uses features from the structural, lexical and  $\alpha$ -equivalence groups and another that also incorporates a semantic component to its scoring function.

The systems I will present in the upcoming section are designed to bring out such differences. My tree matching models, while sharing the same functions for  $\otimes$  (the max function) and  $\oplus$  (a

weighted sum), differ in (a) how the sub-scores above are computed and (b) how the sub-scores are aggregated and transformed to score individual pairs.

## 6.4 Retrieval Models

In this section I describe a series of joint retrieval models, including a typed version of Tangent that is introduced here for the first time (section 6.4.3). The best of each set of models presented here will take part in my evaluation of typed and untyped retrieval in chapter 7.

Most of this section is devoted to explaining my typed tree matching algorithms (section 6.4.1), which should hopefully result in better MIR performance than Tangent’s. I have also designed a set of non-typed tree matching algorithms (section 6.4.2).

All models described in this section score document  $D$  with respect to query  $Q$  using a linear combination of the formula score  $M(Q, D)$  and the text score  $T(Q, D)$  obtained from arbitrary textual models (e.g., the bag-of-words models described in Chapter 2):

$$\alpha \times T(Q, D) + (1 - \alpha)M(Q, D) \tag{7}$$

The difference between Standard Tangent and Typed Tangent is that in the former no type-induced dependence exists between the  $M(Q, D)$  and  $T(Q, D)$ , whereas the latter, like all typed models presented here, links  $T(Q, D)$  and  $M(Q, D)$  via variable typing, type disambiguation and SLT typing from Section 6.3.

In practical typed retrieval  $T(Q, D)$  is computed on text with all formulae removed. Not doing so would amount to double-indexing of some formulae as flat text<sup>7</sup>. More importantly, we do not want any information from the symbolic context to influence the textual context because, by definition, a formula-based MIR model is a more specialised approach to modelling symbolic mathematics and should represent formulae in a more meaningful way than flat text.

In the next section I will introduce two tree matching models that both that make use of type unification in their semantic sub-score and that differ in the way they use the extracted features to produce a score  $M(Q, D)$ <sup>8</sup> for the symbolic modality of document  $D$  with respect to query  $Q$ .

### 6.4.1 Tree Matching with Type Unification

Recall from Section 6.3.3 that (a) my tree matching models compute one sub-score for each kind of similarity identified in Section 6.3.3 using features from the corresponding feature group and (b) set  $\otimes \equiv \max$  to produce  $score(QF_i|D)$  and  $\oplus$  to a weighted sum for producing  $M(Q, D)$ .

**Tree Matching Model 1 (TM1)** My first tree matching model, TM1, computes three exact matching and three subcomponent matching sub-scores for each pair  $(QF_i, QD_j)$ . The first

<sup>7</sup>For example, it is hard to predict how symbolic content will be treated by a textual tokeniser. It might treat "x+y" as a single word, or as three words "x", "+" and "y" (or anything in between, depending on tokeniser settings). Furthermore, more complex formulae can only be partially expressed in ASCII.

<sup>8</sup>Most differences come from the way components for each similarity group are computed and combined per pair, i.e. the function  $score(QF_i, DF_j)$ , that is part of  $M(Q, D)$ .

exact matching sub-score is  $structural_{(QF_i, DF_j)}$ :

$$structural_{(QF_i, DF_j)} = harmonic\_mean \left( \frac{h\_matched}{sNodes}, \frac{h\_matched}{h\_total} \right)$$

The syntactic score is the harmonic mean of the query formula match accuracy (proportion of nodes matched to  $s$  from  $t$ ) and overall match accuracy (proportion of nodes matched to total number of nodes compared). I use  $structural_{(QF_i, DF_j)}$  to measure structural similarity between two formulae, instead of the  $\alpha$ -equivalence feature `isMatch` from Table 6.1, because  $structural_{(QF_i, DF_j)}$  is a continuous score. Scoring structural similarity in proportion to a binary feature would severely penalise structurally similar but not identical formula pairs.

The second sub-score is the lexical score:

$$lexical_{(QF_i, DF_j)} = harmonic\_mean \left( \frac{h\_idmatch}{idS}, \frac{h\_idmatch}{h\_idtotal} \right)$$

The lexical score is the harmonic mean of (a) the proportion of identifiers (i.e., variables) matched over all identifiers query formula  $s$  and (b) the number of comparisons that resulted in a match over the total number of comparisons performed. The score is a balance between the recall of matched identifiers (in isomorphic locations of the two trees) and overall identifier match accuracy.

The third sub-score  $semantic_{(QF_i, DF_j)}$  measures the number of typed nodes that either match directly or can be matched via typed unification (cf. section 6.2):

$$semantic_{(QF_i, DF_j)} = harmonic\_mean \left( \frac{unified}{sTypable}, \frac{unified}{h\_typedtotal} \right)$$

where  $unified$  is the sum of  $h\_typedmatch$ ,  $h\_typedsuffix$  and  $h\_typedexp$ . The score  $semantic_{(QF_i, DF_j)}$  is designed to model similarity based on the denotations assigned to the variables of query and document formulae. This is a more flexible alternative to lexical matching because even if two structurally similar formulae use different denotations for variables in the same position, these may be unified using typed unification.

TM1 also compiles three sub-scores for recursive similarity. These follow the same patterns as their respective direct similarity sub-scores, with  $rec\_unified$  this time defined as the sum of  $rec\_typed\_matched$ ,  $rec\_typed\_suffix$  and  $rec\_typed\_exp$ .

The first of these is the structural recursive score:

$$structural_{R(QF_i, DF_j)} = harmonic\_mean \left( \frac{rec\_matched}{sNodes \times nMatched}, \frac{rec\_matched}{rec\_total} \right)$$

The first argument to the `harmonic_mean` function in  $structural_{R(QF_i, DF_j)}$  measures recursive node recall: the number of recursively matched nodes over the total number of  $s$ -nodes that are known to exist in  $t$ . The second argument measures the recursive accuracy: the aggregate of recursively matched nodes over all comparisons. This term of the score penalises matching larger document expressions (the larger  $t$  is, the more dissimilar it is to  $s$ ) because the number of recursive calls increases and the number of comparisons performed becomes larger.

The second and third are the lexical and semantic recursive scores:

$$lexical_{R(QF_i, DF_j)} = harmonic\_mean \left( \frac{rec\_idmatch}{idS \times nMatched}, \frac{rec\_idmatch}{rec\_idtotal} \right)$$

$$semantic_{R(QF_i, DF_j)} = harmonic\_mean \left( \frac{rec\_unified}{sTypable \times nMatched}, \frac{rec\_unified}{rec\_typed\_total} \right)$$

The sub-scores  $lexical_{R(QF_i, DF_j)}$  and  $semantic_{R(QF_i, DF_j)}$  extend the lexical and semantic components (respectively) to recursive matches of  $s$  to  $t$  (and vice versa) in the same manner as  $structural_{R(QF_i, DF_j)}$  extends  $structural_{(QF_i, DF_j)}$ .

TM1 aggregates the sub-scores by computing their mean, producing a new score for each pair  $(QF_i, DF_j)$ . Following the architecture in Figure 6.28, the score for query formula  $QF_i$  given document  $D$ ,  $score(QF_i|D)$ , is produced by aggregating the scores of  $(QF_i, DF_j)$  pairs by selecting the pair with the maximum score (i.e.,  $\otimes \equiv \max$ ).

The final score  $M(Q, D)$  for  $Q$  with respect to document  $D$  is the weighted sum of the scores for each individual query formula (i.e., the  $\oplus$  function in Figure 6.28):

$$M(Q, D)_1 = \sum_i weight(QF_i, D) \times score(QF_i|D)$$

where

$$weight(QF_i, D) = \frac{\sum_{i,j} nMatched(QF_i, DF_j)}{\sum_{i,j} nTotal(QF_i, DF_j)}$$

**Tree Matching Model 2 (TM2)** My second tree matching model, TM2, is the same as TM1 with a  $\sinh$  transform applied in the last step of my architecture:

$$M(Q, D)_2 = \sinh(\text{score}(Q, D)_1)$$

The  $\sinh$  function introduces larger spread to values closer to 1. I use this property to obtain better differentiation between pairs that are very similar, but too close to 1 to significantly alter the final score otherwise.

#### 6.4.2 Tree Matching without Type Unification

Semantic similarity between two formulae, one coming from the query and another from a document, is a potentially strong signal for mathematical retrieval. In this work type unification is my main contribution to semantic similarity. Recall from the preceding sections that my tree matching models score typed SLT pairs by computing a component for each kind of similarity (syntactic, lexical, semantic and recursive). Models that use typed unification use type unification features, such as `h_typedmatch`, `h_typedsuffix` and `h_typedexp` (cf. Table 6.1 in Section 6.3.3), in their semantic similarity components. However, type unification is a strict form of semantic similarity: it expects two SLTs to be syntactically equivalent in order to reward their semantic overlap. Semantic similarity between two SLTs can also be quantified with relaxed assumptions of syntactic equivalence, without the need to unify the types of variables at identical locations in two SLTs. Models based on the JE method do not attempt to unify the types of constituents at the same positions of formulae (i.e., do not use the aforementioned type unification features). Instead, the JE method computes a semantic score by treating the types in typed SLTs as sets: the semantic overlap score is independent of formula structure. My goal in designing the tree matching models  $\text{TM1}_{JD}$  and  $\text{TM1}_{JE}$  is to establish whether (expensive and relatively strict) type unification can be simulated by far simpler operations that also look at the set of types associated with the SLTs. Differences observed between these models and TM1 can inform me on the feasibility of this proposition.

**Tree Matching Model  $\text{TM1}_{JD}$**  In this version of TM1, type unification (used by the *semantic* sub-scores) is replaced by  $\text{semantic}_{(QF_i, DF_j)_{JD}}$  – a shallow, set-based similarity that is a function of the sets of types occurring in two SLTs. Similarity between query formula  $s$  and document formula  $t$  is computed using the Jacquard coefficient:

$$\begin{aligned} \text{semantic}_{(s,t)_{JD}} &= \text{Jaccard}(\text{types}(s), \text{types}(t)) \\ &= \frac{|\text{types}(s) \cap \text{types}(t)|}{|\text{types}(s) \cup \text{types}(t)|} \end{aligned}$$

where  $\text{types}(s)$  is the set of types in formula  $s$ . The set of types for a formula is extracted by enumerating the types assigned to its typed variables and typable sub-expressions.

**Tree Matching Model TM1<sub>JE</sub>** This version of TM1 extends semantic overlap similarity to types expanded from the embedding space. The *expanded Jaccard semantic overlap* component,  $semantic_{(QF_i, DF_j)_{JE}}$ , is computed over the sets of types for formulae  $s$  and  $t$  that have been expanded using the type embedding space:

$$\begin{aligned} semantic_{(s,t)_{JE}} &= Jaccard(embedexp(types(s)), types(t)) \\ &= \frac{|embedexp(types(s)) \cap types(t)|}{|embedexp(types(s)) \cup types(t)|} \end{aligned}$$

The function *embedexp* takes a set of types as input and uses the type embedding space (Section 4.3) to expand each element in the set with the top 5 most similar types not already in the set. This way, models can simulate semantic-based query expansion, using types, at the formula level. When compared to TM1<sub>JD</sub>, this model evaluates whether types coming from the embedding space will improve the overall score of a formula pair.

**Tree Matching Model TM1-** I also created a version of TM1 that does not include the *semantic* and *semantic<sub>R</sub>* components, i.e. operates without type unification. TM1- is useful when compared against TM1 to quantify the contribution of types and type unification in tree matching.

**Variants of TM2 without typed unification** I also produced the parallel variants for TM2 in the same manner as for TM1, i.e., TM2-, TM2<sub>JD</sub> and TM2<sub>JE</sub>.

### 6.4.3 Typed Tangent Retrieval Models

The Tangent system represents heuristic MIR models in my experiments and is intended to contrast the tree matching formula retrieval models (described in the previous section) that are strict but precise in their quantification of formula similarity. I consider them precise because they compare sub-tree by sub-tree, and strict because they penalise the smallest structural difference. I have implemented Tangent indexing and scoring<sup>9</sup> along with extensions for typed formula retrieval, resulting in the following two Tangent systems:

**Untyped Tangent (UT).** This model represents out-of-the-box Tangent retrieval as described by Pattaniyil and Zanibbi (2014):  $T(d)$  takes the form of Lucene VSM text retrieval and  $M(d)$  is obtained using Tangent scoring over a Tangent raw symbol tuple index (Section 2.2.5).

**Typed Tangent (TT).** This model is a typed variant of Tangent, introduced here for the first time. To realise this model, tangent formula indexing is performed as described in Section 2.2.5, except that it uses typed instead of untyped SLTs. In the resulting typed Tangent index formulae are approximated by tuples with types in place of raw symbols. SLT variable nodes that have not

<sup>9</sup>Note that I have not implemented Tangent’s wildcard scoring. One of the reasons for this decision is that the naturally observed queries in my test collection (introduced in chapter 3) do not contain any wildcards; another, more philosophical reason concerning query language design is described on page 61.

been annotated by any type are represented by their raw symbol in this typed index. Recall the example from the beginning of this chapter: the formulae  $hom(G_1, G_2)$  and  $hom(R_1, R_2)$ , with  $G_1, G_2 :: GROUP$  and  $R_1, R_2 :: RING$ . The resulting typed Tangent index for these formulae is shown in Table 6.3; for comparison, the corresponding untyped Tangent index for the formulae can be found in Table 6.1 (page 128).

| $hom(G_1, G_2)$ |              |       |       | $hom(R_1, R_2)$ |             |       |       |
|-----------------|--------------|-------|-------|-----------------|-------------|-------|-------|
| Parent          | Child        | Dist. | Vert. | Parent          | Child       | Dist. | Vert. |
| hom             | (            | 1     | 0     | hom             | (           | 1     | 0     |
| hom             | <i>GROUP</i> | 2     | 0     | hom             | <i>RING</i> | 2     | 0     |
| hom             | ,            | 3     | 0     | hom             | ,           | 3     | 0     |
| hom             | <i>GROUP</i> | 4     | 0     | hom             | <i>RING</i> | 4     | 0     |
| hom             | )            | 5     | 0     | hom             | )           | 5     | 0     |
| (               | <i>GROUP</i> | 1     | 0     | (               | <i>RING</i> | 1     | 0     |
| (               | ,            | 2     | 0     | (               | ,           | 2     | 0     |
| (               | <i>GROUP</i> | 3     | 0     | (               | <i>RING</i> | 3     | 0     |
| (               | )            | 4     | 0     | (               | )           | 4     | 0     |
| <i>GROUP</i>    | ,            | 1     | 0     | <i>RING</i>     | ,           | 1     | 0     |
| <i>GROUP</i>    | <i>GROUP</i> | 2     | 0     | <i>RING</i>     | <i>RING</i> | 2     | 0     |
| <i>GROUP</i>    | )            | 3     | 0     | <i>RING</i>     | )           | 3     | 0     |
| ,               | <i>GROUP</i> | 1     | 0     | ,               | <i>RING</i> | 1     | 0     |
| ,               | )            | 2     | 0     | ,               | )           | 2     | 0     |
| <i>GROUP</i>    | )            | 1     | 0     | <i>RING</i>     | )           | 1     | 0     |
| )               | None         | 0     | 0     | )               | None        | 0     | 0     |

Table 6.3: Typed Tangent formula index entries for  $hom(G_1, G_2)$  and  $hom(R_1, R_2)$ .

Having described these typed joint models, I now have all the pieces necessary to empirically test both of my hypotheses concerning MIR, which I will do in the next chapter.

## 6.5 Chapter Summary

In this chapter, I argued that modelling text and formulae jointly via types is a novel idea in MIR. The main idea in typed retrieval is typed unification (Section 6.2), which allows syntactically equivalent expressions to be unified based on the types of their constituents, rather than their raw symbols.

My second hypothesis is that retrieval of formulae (represented by SLT trees) using typed unification will improve MIR when compared to treating either modality on its own. Typed unification should result in an improvement because it hopefully enables models to better quantify semantic similarities between formulae, as discussed at the beginning of this chapter.

Typed unification is supported by variable typing (Chapter 5), by type disambiguation (Section 6.3.1) and by the typing of non-variable SLTs (Section 6.3.2). Chaining these procedures

together establishes the link between modalities required by typed unification, with the link taking the form of typed SLTs.

In Section 6.3.3 I introduced the general design of my tree matching retrieval models: scoring the symbolic modality of a document with respect to a query involves finding the document that is most similar to each query formula (i.e., the function  $\otimes$ ) and aggregating the individual formula pair scores (i.e., the function  $\oplus$ ).

Computing document–query formula similarities (the  $score(QF_i, DF_j)$  function from Section 6.3.2) is done using a set of sub-scores, which correspond to each kind of similarity identified in Section 6.3.3). The sub-scores contribute independently to the similarity of formula pairs before aggregation. This approach increases the robustness of my tree matching models in cases where information is missing and allows me to design controlled experiments in a flexible manner.

Finally, in Section 6.4, I described how the typed retrieval principles translate into models that can be experimentally evaluated on my test collection. I define here, out of a myriad of possible typed tree matching systems, two models, TM1 and TM2, both of which use type unification and differ only in the way they score document–query formula pairs. I also present a typed Tangent-based baseline (first introduced here). Running against these strict typed tree matching models I also introduced two shallower typed models which replace expensive type unification with Jaccard-style semantic similarity (with and without type expansion).

In the upcoming chapter, I will present experiments on models that explore the influence of different usages of type information from chapters 4 to the current chapter 6:

- textually-typed models such as `Types2X` and `Types2XExp` that use types only as textual signals, from Section 4.4.
- fully untyped models that use only formula retrieval, from Sections 2.2.3 and 2.2.5, such as `UT` and the tree matching models introduced in Section 6.4.1 with typing switched off, i.e., `TM1-` and `TM2-`.
- models that jointly model text and formulae via types, divided into
  - those that use type unification (`TM1` and `TM2`; section 6.4.1)
  - and shallow-typed models without unification (`TT`, `TM1JD`, `TM1JE`, `TM2JD` and `TM2JE`; section 6.4.2).

# Chapter 7

## Retrieval Experiments

In previous chapters I introduced ideas and work that are prerequisites to my experimental investigation of the two main hypotheses of this thesis.

In section 7.1, I will investigate hypothesis 1, namely that treating types as atomic units of information, rather than a bag-of-words, will improve efficiency in textual models. The models I built to enable this investigation are the Types2X, Types2XExp and TFIDF-Exp retrieval models described in Chapter 4.

Section 7.2 serves to establish some baselines for pure formula retrieval, which are useful when interpreting the experiments testing my second hypothesis. My second hypothesis asks whether the models from Chapter 6, which use types extracted from the text as denotations to symbols in formulae, have better retrieval efficiency than those that do not link text and formulae but treat them independently.

To this end, section 7.2 establishes baselines for type agnostic text and untyped formula retrieval for tree matching models TM1 and TM2, and for Tangent.

Section 7.3 then tests hypothesis 2 itself, by evaluating the type-aware versions of TM1, TM2 and Tangent, against their untyped counterparts.

I will now discuss a practical consideration for the evaluation of the retrieval models presented in Sections 7.2 and 7.3, which concerns the training and test corpora used. My test collection (which I call  $\mathbf{Q}$ ) contains 160 queries, but only 116 of these contain any formulae (Section 3.2.5). This subset is called  $\mathbf{Q}_F$ .

But not all of these formulae are usable with typed retrieval; for instance, if a query doesn't contain a single typable variable, typed retrieval could not possibly make any difference to the performance of a system with regards to this query. In my test collection, only 106 queries fulfil the prerequisite for the meaningful evaluation of typed retrieval, in that the query contains at least one typable formula. This is the subset of the corpus I call  $\mathbf{Q}_{FT}$ ; I will be using it in Section 7.3. In the context of typed retrieval, it would be less meaningful to use either  $\mathbf{Q}$  or  $\mathbf{Q}_F$ , as the former may contain queries with no formulae at all, whereas the latter, where all queries do contain formulae, may still contain some without a single typable variable.

Section 7.2 deals with untyped models, where it does not matter whether the formulae can be typed or not; the most appropriate query subset to use for these results is therefore  $\mathbf{Q}_F$ . Performance metrics compiled with  $\mathbf{Q}$  would contain interference from formula-free queries, whereas performance metrics compiled with  $\mathbf{Q}_{FT}$  cover only a subset of the cases where untyped models have a potential advantage<sup>1</sup>.

Table 7.1 summarises the three subsets of the corpus I will be using in the three sections of this chapter.

| Subset            | No. of queries | Description  |
|-------------------|----------------|--|
| $\mathbf{Q}$      | 160            | All queries: may or may not contain formulae; may or may not contain textual type mentions, typed variables or typable formulae. |
| $\mathbf{Q}_F$    | 116            | Those queries that contain formulae; may or may not contain textual type mentions.   |
| $\mathbf{Q}_{FT}$ | 106            | Those queries that contain formulae with at least one typed variable identified by my preprocessing in Section 5.4.              |

Table 7.1: Query subsets with properties and query numbers.

## 7.1 Type-based Textual Retrieval

My first hypothesis is that textual mathematical types are important discriminators in the mathematical discourse. Furthermore, I hypothesise that semantic relationships between mathematical types can be exploited to obtain significant improvements in retrieval efficiency of mathematical information. In Section 4.4 I proposed two type-based heuristic models that assign elevated significance to types: `Types2X` and `Types2XExp`. The `Types2XExp` model also makes use of inter-type similarity, encoded in a type embedding space, to expand queries. This work was published as Stathopoulos and Teufel (2015).

### 7.1.1 Experimental Design

I adopted a two-level comparison of retrieval models. On one level, I compare traditional, term-based retrieval models to type-aware derivatives. At this level, I wish to (a) determine the performance of traditional IR models on the CUMTC and (b) investigate if types are more useful than simple terms when retrieving research-level mathematics. To do so, the first level of comparison is performed across IR paradigms; it includes term-based models that employ heuristic methods (e.g., VSM and BM25) as well as language modelling (e.g., classical multinomial language model (Zhai and Lafferty, 2001)).

On the second level, I compare term-based query expansion with type-based query expansion. My `Types2XExp` textual model performs query expansion using types. In case its performance is good, it is possible that this is due to the fact that it employs query expansion, irrespective of its

<sup>1</sup>For completeness and strict comparability, Appendix D will nevertheless present results on subsets and tuning methods not reported in the main text.

treatment of types. To account and control for this eventuality, my evaluation includes versions of language IR models augmented with state-of-the-art query expansion based on pseudo-relevance feedback (PRF): RM3 (Abdul-jaleel et al., 2004; Lv and Zhai, 2009) and PRM2 (Lv and Zhai, 2011). Successful performance of Types2XExp could also be due to the fact that its query expansion uses an embedding space of any kind, irrespective of the fact that its embedding space is based on types. I test against this eventuality by replacing the typed embedding space for QE from Types2XExp with a simpler term-based embedding space. In this model, called TFIDF-Exp, VSM is used as the base model, and the query is expanded using the top  $s$  terms in the query as determined by document collection-wide TF-IDF scores.

|              | No Expansion                    | Query Expansion |
|--------------|---------------------------------|-----------------|
| <b>Terms</b> | Heuristic/Probabilistic IR:     |                 |
|              | VSM                             | TFIDF-Exp       |
|              | BM25                            | -               |
|              | Language model-based IR:        |                 |
|              | $MLM_{jm}$                      | $MLM_{jm}+RM3$  |
| $MLM_{dir}$  | $MLM_{dir}+RM3, MLM_{dir}+PRM2$ |                 |
| $MLM'_{dir}$ | $MLM'_{dir}+RM3$                |                 |
| SPUD         | SPUD+RM3, SPUD+PRM2             |                 |
| <b>Types</b> | Types2X                         | Types2XExp      |

Table 7.2: Overview of models.

I will now describe the retrieval models in this experiment, which are also listed in Table 7.2.

**Type-aware models.** My two type-aware models Types2X and Types2XExp are both extensions of Lucene’s VSM implementation and include a two-fold up-weighting component applied to types. The difference between them is that Types2X does not use query-expansion and Types2Exp does, by modeling type similarity in an embedding space.

**Types2X** **Lucene VSM with 2X type boosting.** Types2X, described in Section 4.4, is used to measure the simplest possible way of incorporating types during retrieval (as opposed to just using terms).

**Types2XExp** **Lucene VSM with 2X type boosting and type-based query expansion.** Described in Section 4.4, Types2XExp is based on Types2X and is designed to investigate the effects of type-based query expansion. Types2XExp enriches queries with types in order to deal with type-sparse queries. Queries are expanded using the types (as opposed to the terms) they contain.

All other models evaluated here are baselines of different kinds and complexities. **Traditional models** that do not use query expansion are used as baselines intended to identify and quantify the effects of boosting types. I consider both heuristic/probabilistic models (e.g., VSM, BM25)

and models based on language modeling, which are naturally extensible with sophisticated query expansion techniques:

**VSM** I used Lucene’s default VSM implementation, both as a baseline but also as the basis of some of my models. Lucene’s scoring function is an instance of cosine similarity.

**BM25** I used Lucene’s BM25 implementation with default parameters.

**Language model-based IR models.** These language models are included as more sophisticated, type-agnostic alternatives to the basic heuristic models. I used three models, SPUD, and MLM with two different kinds of smoothing, Jelinek-Mercer and Dirichlet smoothing<sup>2</sup>. MLM is a well-known language model, and SPUD is included in my comparison as a state-of-the-art type-agnostic LM baseline against which I can benchmark the performance of my type-based models.

Two versions of Dirichlet smoothing are used because of a suspected software error in the Lucene implementation ( $LM_{dir}$ )<sup>3</sup>.

**MLM<sub>jm</sub>** For the multinomial model with JM smoothing, I use Lucene’s default implementation. In the absence of training data for the parameter  $\alpha$ , I use  $\alpha = 0.7$ , since there is strong evidence that this value is optimal for long queries (Zhai and Lafferty, 2001; Zhai and Lafferty, 2004), as is the case in my setup. The smoothing parameter,  $\mu$ , is set to  $\mu = 2000$ , which Zhai and Lafferty (2001, 2004) found to be near-optimal for large queries, such as those in my setup.

**MLM’<sub>dir</sub>** The LUCENE implementation of the multinomial language model with Dirichlet smoothing (suspected to be problematic).

**MLM<sub>dir</sub>** The multinomial language model with Dirichlet smoothing (implementation by (Cummins et al., 2015)).

**SPUD** I use the SPUD implementation by Cummins et al. (2015) with default parameters ( $\omega = 0.8$ ), which has been shown to produce good results with long queries (Cummins et al., 2015).

**Term-based query expansion.** This is a VSM model with a very simple term-based form of query expansion using a raw term embedding space. This baseline is used to determine whether any performance improvements obtained by type-based QE using an embedding space are due to types, rather than the use of embedding spaces in general.

---

<sup>2</sup>I used implementations by Cummins et al. (2015) (<https://github.com/ronancummins/spud>) for all language models tested here.

<sup>3</sup>In correspondence with one of the authors of Cummins et al. (2015), it has come to my attention that there is suspicion in the community that the Lucene implementation of the classical multinomial language model may be incorrect for large queries, so I report two implementations for safety.

**TFIDF-Exp** This model performs query expansion using a fixed number of important terms in the query. TF-IDF for this model is calculated using PyLucene<sup>4</sup>, a Python interface to Lucene. Stopwords and words with term frequency lower than 50 are excluded. Like in Types2XExp, each selected term is expanded using its  $n$ -nearest neighbours in a word embedding space (skip-gram, window size =10). The model has two parameters: *pool size* ( $n$ ) and *seed size* ( $s$ ). I found experimentally on my development set that the model performs best for  $n = 1$  and  $s = 1$ .

### Established query expansion models

I tested the two query expansion models described in Section 2.1, RM3 and PRM2, in combination with the language models described above, resulting in 6 systems as follows<sup>5</sup>:

**MLM<sub>jm</sub>+RM3** The multinomial language model with JM smoothing and RM3 Query Expansion.

**MLM'<sub>dir</sub>+RM3** The LUCENE implementation of the multinomial language model with Dirichlet smoothing and RM3 Query Expansion (suspected to be problematic).

**MLM<sub>dir</sub>+RM3** The multinomial language model with Dirichlet smoothing and RM3 Query Expansion (implementation by Cummins et al. (2015)).

**SPUD+RM3** SPUD with RM3 Query Expansion

**MLM<sub>dir</sub>+PRM2** The multinomial language model with Dirichlet smoothing and PRM2 Query Expansion.

**SPUD+PRM2** SPUD with PRM2 Query Expansion

Figure 7.1 shows the relationships between basic models and their more sophisticated derivatives, where white and grey boxes represent term-based and type-based models, respectively.

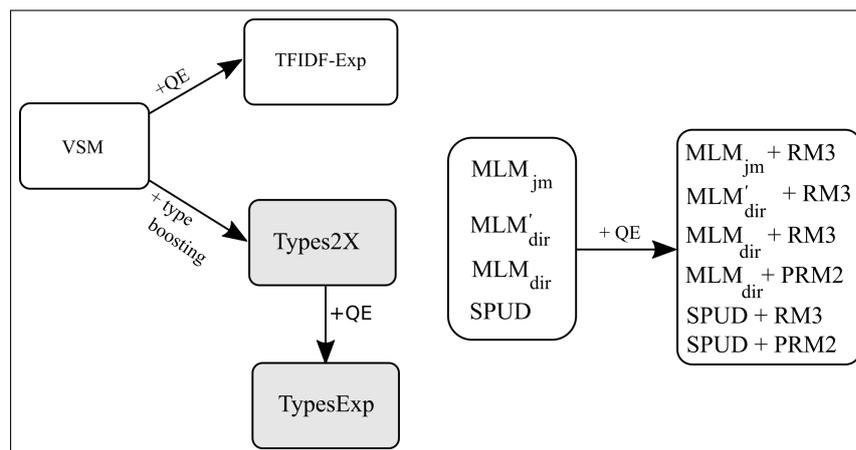


Figure 7.1: Derivational relationship between IR models evaluated in Section 7.1.

<sup>4</sup><https://lucene.apache.org/pylucene/>

<sup>5</sup>Not all combinations of language models and query expansion methods were technically feasible.

### 7.1.2 Results and Discussion

Table 7.3 shows the results of all retrieval models in Mean Average Precision (MAP). The table is organised into three groups. Model MAPs are presented in the first row of each group. The last two rows in each group indicate the significance in MAP difference between each model (column) and the Types2XExp and Types2X models. Significance is obtained using the permutation test (Section A.3) and reported here as follows:  $\ll$  indicates that "column" is significantly better than "row" at significance level  $\alpha = 0.01$ ;  $<$  at  $\alpha = 0.05$ .  $\approx$  indicates that the difference is not significant. Table 7.4 reports significance between models not employing query expansion, and Table 7.5 shows the difference between unexpanded and query-expanded term-based language models.

|        | VSM                      | BM25      | MLM <sub>jm</sub>      | MLM' <sub>dir</sub>      | MLM <sub>dir</sub>      |
|--------|--------------------------|-----------|------------------------|--------------------------|-------------------------|
| MAP    | .076                     | .077      | .084                   | .072                     | .066                    |
| T2XExp | $\gg$                    | $\gg$     | $\gg$                  | $\gg$                    | $\gg$                   |
| T2X    | $\approx$                | $\approx$ | $\approx$              | $\approx$                | $\approx$               |
|        | SPUD                     | TFIDF-Exp | MLM <sub>jm</sub> +RM3 | MLM' <sub>dir</sub> +RM3 | MLM <sub>dir</sub> +RM3 |
| MAP    | .090                     | .060      | .063                   | .051                     | .082                    |
| T2XExp | $\gg$                    | $\gg$     | $\gg$                  | $\gg$                    | $\gg$                   |
| T2X    | $\approx$                | $\approx$ | $\approx$              | $\approx$                | $\approx$               |
|        | MLM <sub>dir</sub> +PRM2 | SPUD+RM3  | SPUD+PRM2              | Types2X                  | Types2XExp              |
| MAP    | .061                     | .050      | .072                   | .070                     | .173                    |
| T2XExp | $\gg$                    | $\gg$     | $\gg$                  | $\gg$                    | -                       |
| T2X    | $\approx$                | $\approx$ | $\approx$              | -                        | $\ll$                   |

Table 7.3: Model MAP performance compared to Types2XExp (T2XExp) and Types2X (T2X) models.

|                     |           |           |                   |                     |                    |           |      |
|---------------------|-----------|-----------|-------------------|---------------------|--------------------|-----------|------|
| VSM                 | -         |           |                   |                     |                    |           |      |
| BM25                | $\approx$ | -         |                   |                     |                    |           |      |
| MLM <sub>jm</sub>   | $\approx$ | $\approx$ | -                 |                     |                    |           |      |
| MLM' <sub>dir</sub> | $\approx$ | $\approx$ | $\approx$         | -                   |                    |           |      |
| MLM <sub>dir</sub>  | $\approx$ | $\approx$ | $\approx$         | $\approx$           | -                  |           |      |
| TFIDF-Exp           | $\approx$ | $\approx$ | $\approx$         | $\approx$           | $\approx$          | -         |      |
| SPUD                | $\approx$ | $\approx$ | $\approx$         | $\approx$           | $>$                | $\approx$ | -    |
|                     | VSM       | BM25      | MLM <sub>jm</sub> | MLM' <sub>dir</sub> | MLM <sub>dir</sub> | TFIDF-Exp | SPUD |

Table 7.4: Comparison of models not employing query expansion.

As expected, absolute MAPs are low across the board, a phenomenon that can be attributed to the complexity of the underlying information needs and the resulting small number of relevant documents per query. But as long as the evaluation is stable, it is the comparative performance of each model that we should be interested in.

The model utilising type-based expansion (Types2XExp) is the best model at MAP=0.173;

it outperforms every other model<sup>6</sup>. In some cases the differences are dramatic; Types2XExp’s MAP score is twice that of the Lucene VSM.

I will now discuss which of Types2XExp’s components contributed most to this improvement over existing IR models. The difference in MAP between Types2X, which up-weights types on its own (MAP=0.07) and VSM (MAP=0.76) is not statistically significant. This is also the case when comparing type up-weighting (Types2X) to the majority of alternative models.

The best-performing model not employing query expansion is SPUD (MAP=0.90), followed by MLM with JM smoothing (MAP=0.84). The traditional BM25, VSM and Lucene  $MLM'_{dir}$  models performed comparably to each other (MAP= 0.77, 0.76, and 0.72), with no significant differences in MAP between them, as shown in Table 7.4.

I now shift my attention to models using term-based query expansion. Table 7.5 records the difference in performance of the four language models considered, when query expansion by RM3 or PRM2 is added (\*\* stands for significance at 0.01; \* for significance at 0.05). We can observe that versions of the models augmented with state-of-the-art query expansion are outperformed by their corresponding basic versions, often significantly. General-purpose query expansion methods appear to be ineffective in retrieving research-level mathematics. This is in contrast to type-based query expansion, where the Types2XExp model significantly outperforms both the VSM and Types2X models it is based on.

|       | $MLM_{jm}$     | $MLM'_{dir}$    | $MLM_{dir}$ | SPUD            |
|-------|----------------|-----------------|-------------|-----------------|
| +RM3  | <b>-0.021*</b> | <b>-0.021**</b> | -0.016      | <b>-0.040**</b> |
| +PRM2 | —              | —               | -0.005      | -0.019          |

Table 7.5: Comparison of unexpanded and query-expanded term-based language models.

TFIDF-Exp performed worse than the VSM baseline it is based on (.06 vs .076 MAP) and was also outperformed by the Types2X (.07 MAP), which does not use query expansion in any form, but the differences in both cases are not significant.

We have seen this pattern in previous paragraphs: query expansion based on terms often has a detrimental effect in retrieving research-level mathematics; the task overall benefits more from term-based models with no query expansion at all. The idea of simply up-weighting types (Types2X, .07 MAP) yields better results in MAP than term-based query expansion used in TFIDF-Exp (.06 MAP),  $MLM_{dir}$ +PRM2 (.061 MAP),  $MLM_{jm}$ +RM3 (.063 MAP) and  $MLM'_{dir}$ +RM3 (.051 MAP)<sup>7</sup>.

These observations seem to point to the fact that, in the context of MIR, mathematical types encode more information than the sum of their individual, constituent terms. When choosing which atomic units to use in textual MIR, types seem to be a better choice than individual words,

<sup>6</sup>Both Type2XExp and Types2X models show higher performance than the results I published in (Stathopoulos and Teufel, 2016). This is because at the time of publication, I had not yet expanded the type seed dictionary and was therefore working with 10,601 types rather than 1.23 million types. Appendix D.1.1 shows both results tables for comparison.

<sup>7</sup>the differences are not significant.

thus confirming hypothesis 1.

On one hand, the performance of Types2X suggests that information coming from the types occurring in the queries alone may not be enough to produce significant improvements in retrieval efficiency, where queries are overall difficult. On the other hand, it is only the combination of query expansion with type information (rather than with simple terms) that yields significant performance gains.

My intuition is that type-based expansion introduces semantically related concepts that elevate the score of topically relevant documents. Insight into why this method performs well can be obtained by looking into how types are expanded. The query in Table 7.6 is topic 175 (MO post 90038<sup>8</sup>). From an initial set of 6 types found in the query, my method expanded the query with 14 more types.

|                |   |
|----------------|---|
| Topic          | Let $P$ be a <i>parabolic subgroup</i> of $GL(n)$ with <i>Levi decomposition</i> $P = MN$ , where $N$ is the <i>unipotent radical</i> . Let $\rho$ be an <i>irreducible representation</i> of $M(Z_p)$ inflated to $P(Z_p)$ , how does $Ind_{P(Z_p)}^{GL_n(Z_p)} \rho$ decompose? It would be sufficient for me to know the result in the simplest case, where $P$ is a <i>Borel subgroup</i> . |
| Query Types    | 'levi decomposition', 'parabolic subgroup', 'unipotent', 'irreducible representation', 'borel subgroup'   |
| Expanded Types | 'maximal parabolic subgroup', 'unitary irreducible representation', 'minimal parabolic subgroup', 'irreducible corepresentation', 'levi subgroup', 'lie subalgebra', 'parabolic subalgebra', 'levi factor', 'maximal torus', 'reducible representation', 'maximal unipotent subgroup', 'irreducible unitary representation', 'borel subalgebra', 'unipotent subgroup'                           |
| TFIDF-Exp      | 'nilpotent', 'shredded', 'semi-simple', 'inflates', 'centralizer', 'pro-', 'puffed', 'engulfed', 'inflate', 'semisimple'  |

Table 7.6: Types in query (and dictionary), Types expanded by Types2XExp and TFIDF-Exp for Topic 175.

A mathematician would recognise that the overall topic of the query is Lie algebras and Lie groups and appreciate the strong topical connection between the types in the original query and those added by Types2XExp. For example, the type “Levi factor” in the expanded set for query 175 is a “Levi subalgebra” produced from the “Levi decomposition” (query type) of “Lie algebras”. In the example in Table 7.6, our mathematician would also recognise that Types2XExp finds types that are special instances of those in the query. For example, “minimal parabolic subgroup” and “irreducible unitary representation” are special instances of “parabolic subgroup” and “irreducible representation”, respectively. On the other hand, the TFIDF-Exp expansion set for this query (bottom row of Table 7.6) appears less likely to contain terms related to the topic subject.

The “bag-of-types” Types2X model performed unexpectedly badly (.070 MAP), below VSM, although not significantly so. Looking at Topic 175, we see a possible explanation for this underperformance. Types2X simply has nothing to play with – queries are short, the type vocabulary

<sup>8</sup><http://mathoverflow.net/questions/90038>

is sparse, and there is minimal superficial overlap in terms of types between relevant document and query without type expansion. This is in stark contrast to the potential of type-based query expansion (Types2XExp, .173 MAP).

I speculated that the situations where methods that exploit semantic type relatedness, such as Types2XExp, can be advantageous are those where (a) lexical sparsity requires the use of some form of generalisation or similarity across concepts, of which QE is a simple variant, and (b) information about the similarity between types is more informative than similarity between raw words. The qualitative analysis on Topic 175 has given some concrete examples confirming that such improvements might be a common occurrence.

## 7.2 Untyped Formula-Based Retrieval

Let us now turn to a different kind of model altogether, namely one that concentrates on treating the formulae in query and documents. In the previous section, textual retrieval models treated formulae in the text as bag-of-words. In the models presented in this section, formulae are separated from the text and delegated to dedicated formula retrieval methods, whose job it is to index and score the formulae. The remaining text is treated by a text model, which is not of primary interest here.

### 7.2.1 Experimental Design

In the experiments presented here, I compare three formula retrieval models: Tangent, which is a structural heuristic for indexing and retrieval of raw symbols (Section 2.2.5), and the untyped variants TM1– and TM2– of the tree matching models described in Section 6.4.2. All these formula retrieval models are based on raw symbols; in other words they are untyped. In fact, TM1– and TM2– do not have a semantic similarity component in their formula similarity function; instead, they retrieve formulae based only on syntactic and lexical similarity (up to  $\alpha$ -equivalence).

The textual component is a bag-of-words method, either VSM or BM25. Note that these textual models are quite different from the ones I discussed in the previous section. The models from Section 7.1, typed or untyped, are stand-alone bag-of-words models that are unaware of formulae: they treat formulae as text. This is achieved by extracting the elements of MathML trees and treating their context as part of the text. In the current section, formulae are separated from the text; they are *not* treated as text, but explicitly handled by a formula retrieval component, which is the object of study in this section.

Combining every textual model with every formula retrieval model results in the six model variants shown in Table 7.7.

| Name      | Formula  | Text |
|-----------|--|------|
| VSM+UT    | Tangent heuristic formula indexing with raw symbols.           | VSM  |
| BM25+UT   | Tangent heuristic formula indexing with raw symbols.           | BM25 |
| VSM+TM1–  | Tree matching algorithm 1, without any semantic sub-component. | VSM  |
| BM25+TM1– | Tree matching algorithm 1, without any semantic sub-component. | BM25 |
| VSM+TM2–  | Tree matching algorithm 2, without and semantic sub-component. | VSM  |
| BM25+TM2– | Tree matching algorithm 2, without any semantic sub-component. | BM25 |

Table 7.7: System variants with textual and formula components.

In order to allow for comparison to the original Tangent system, I use the same combination method for textual and formula retrieval scores as is used in Tangent, namely linear interpolation:

$$\alpha T(d) + (1 - \alpha)M(d),$$

The parameter  $\alpha$  controls how much weight is placed on the textual component during scoring. In the original Tangent paper,  $\alpha$  was empirically set by a search of the interval between 0 and 0.5. My test collection is different to that used by Pattaniyil and Zanibbi (2014); for instance, the formula-to-token ratio (f-t-t) across the test collection is different (see Table 3.7 in Section 3.2.5). Theoretically there are two ways how one can adapt estimation of  $\alpha$  to my test collection, and I will explore both:

- Estimation of  $\alpha$  by searching for the optimal value on my development set. I automatically search the interval  $[0, 1]$  for the value of  $\alpha$  that maximises the MAP for each model on the development set. I consider 18 equidistant points in the interval in this search. I think that searching the entire range  $[0, 1]$  is the right thing to do, whereas Pattaniyil and Zanibbi (2014) searched only the lower half of that range.
- Estimation of  $\alpha$  using the document collection mean formula-to-token, in my case  $\alpha = 1 - 0.074 = 0.926$ .

In my experiments, I will additionally show results for the case where  $\alpha = 1$ . This configuration disables the formula retrieval component and uses only the textual model; as explained before, this corresponds to a case where all formulae are removed from the input. The performance of no-formula textual models is expected to be easily outperformed by models that additionally incorporate formula retrieval; in other words it is a weak baseline. I nevertheless show the performance of the no-formula models because this allows me to quantify the effect of the additional work done by each formula model. I will present results first for corpus  $\mathbf{Q}_F$ , as this is the most appropriate setting for the interpretation of untyped formula retrieval. Queries that do not contain formula are irrelevant in this setting, and you may remember that in  $\mathbf{Q}_F$  exactly

these queries have been removed. On the other hand, as typing is not of relevance here, those queries which contain formulae but no types remain.

## 7.2.2 Results and Post-hoc analyses

Table 7.8 shows the results in MAP of untyped formula retrieval for all models on  $Q_F$ , under three tuning scenarios:  $\alpha$  tuned on the DevSet,  $\alpha$  tuned with the f-t-t method, and  $\alpha = 1$ , which constitutes the baseline (No-Formula Model). The numbers in brackets in Table 7.8 give the difference between the joint formula model and their no-formula, text only baselines. Boldface numbers indicate that the difference is statistically significant<sup>9</sup>.

| $\alpha$      | Textual model | No-Formula model | TM1–         | TM2–                  | UT                    |
|---------------|---------------|------------------|--------------|-----------------------|-----------------------|
| DevSet tuning | VSM           | .067             | .053 (–.014) | .029 (– <b>.038</b> ) | .088 (+.021)          |
|               | BM25          | .066             | .082 (+.016) | .062 (–.004)          | .068 (+.002)          |
| f-t-t ratio   | VSM           | .067             | .081 (+.014) | .027 (– <b>.040</b> ) | .071 ( <b>+.005</b> ) |
|               | BM25          | .066             | .084 (+.017) | .055 (–.011)          | .065 (–.001)          |

Table 7.8: Performance of untyped Formula Models on  $Q_F$  in MAP, with differences to respective No-Formula models (bold if significant).

I will first consider the methodological question which style of tuning is better. It is the f-t-t method with which the highest significant improvement over baseline was achieved (by the VSM+UT model). On the other hand, we observe that tuning  $\alpha$  on the DevSet seems to produce higher MAPs overall, including the numerically highest MAP observed of .088 (again achieved by the VSM+UT model). However, this is not significantly different from the baseline performance at .067<sup>10</sup>. Thus, since differences between the formula retrieval models and their (no-formula) textual baselines are more pronounced with the f-t-t method, I will focus my discussion on results obtained using this tuning method.<sup>11</sup>

We can see several more things from Figure 7.8. First, even the highest performance of any model in this section (MAP=.084) remains far below the performance of the best type-based textual retrieval model from Section 7.1. Second, models based on TM1– perform consistently better than models based on TM2–: while TM1– models hover numerically around their no-formula textual baselines (sometimes even performing numerically slightly above them, although never significantly so), the performance of TM2– is always numerically worse than its baseline, in the case of VSM-based models even significantly so. This suggests that the application of

<sup>9</sup>I am again using the paired permutation test.

<sup>10</sup>There are some effects around significance testing that look anomalous. For instance, although the difference between VSM+UT and its baseline is small (.005) for the f-t-t method compared to that for the DevSet tuning method (.021), it is significant whereas the DevSet difference is not. In addition, the difference between VSM+TM1– and VSM+UT, both tuned with the f-t-t method, is relatively high (0.14), but the permutation test does not detect any significance. This merits further investigation, but for now I will hypothesise that the aforementioned observations have to do with the fact that MAP is a summary metric: large average precision differences in a few queries push MAP scores higher, but the overall difference is rejected by the permutation test because it lacks consistency. For this reason, I consider significance as a deciding factor in my interpretation and analysis of results.

<sup>11</sup>From here onwards, unless otherwise specified, tuning is always performed using the f-t-t method.

the sinh transform intended to redistribute the scores produced by TM2– actually has a negative effect on the model’s overall performance.

Third, based purely on raw MAPs, we observe that overall BM25+TM1– performs numerically reasonably well regardless of the tuning method, that VSM+UT is a strong model and that BM25 seems to work better than VSM when combined with the TM1– and TM2– formula models.

|            |      |        |          |         |           |           |
|------------|------|--------|----------|---------|-----------|-----------|
| VSM+TM1–   | .081 | ≈      |          |         |           |           |
| BM25+UT    | .065 | ≈      | ≈        |         |           |           |
| BM25+TM1–  | .084 | ≈      | ≈        | ≈       |           |           |
| BM25+TM2–  | .055 | ≈      | ≈        | ≈       | ≪         |           |
| VSM+TM2–   | .027 | ≪      | ≪        | <       | ≪         | ≈         |
| <b>MAP</b> |      | .071   | .081     | .065    | .084      | .055      |
|            |      | VSM+UT | VSM+TM1– | BM25+UT | BM25+TM1– | BM25+TM2– |

Table 7.9: Significance of retrieval models on  $Q_F$  ( $\alpha$  tuned using f-t-t).

Table 7.9 gives significance for each pair of models for the f-t-t  $\alpha$  tuning method<sup>12</sup>. Unfortunately, the direct significance tests between models do not help in distinguishing among the models, except to single out VSM+TM2–’s performance as worse than all others.

I will now perform three post-hoc analyses, investigating two questions:

1. So far, Tangent (VSM+UT) appears to be the strongest model for combined retrieval of text and raw symbol. How does VSM+UT rank documents differently from its baseline, VSM, and from VSM+TM1–?
2. VSM+TM1– is consistently better than VSM+TM2–. What is different about how these models rank relevant documents?

So far, we have compared systems only by MAP, but as a summary metric, MAP masks variations in performance in individual queries. To answer the questions above, we will consider system performance query-by-query, trying to characterise which properties of a query and/or relevant document make one system perform better than another. By looking at retrieval contexts (query/relevant document properties) on those query subsets where one system performs better than the other, we can derive a finer-grained performance profile of a pair of models which can sometimes explain the observed difference in performance between them.

### 7.2.2.1 Performance of UT

Concerning the first question, I will first compare VSM+UT with VSM+TM1–, and later compare it against raw VSM.

<sup>12</sup>For completeness I include significance testing results for tuning with DevSet in Appendix D, Table D.4.



**VSM+UT vs VSM+TM1-.** When comparing the mean average absolute difference in rank between VSM+UT (model A) and VSM+TM1- (model B)<sup>13</sup>, I found it to be around 5,760 positions<sup>14</sup> in favour of VSM+UT, a difference that is significant at level  $\alpha=0.05$ .

I hypothesised that there might be properties of individual queries which might influence the systems' behaviour, and therefore conducted a per-query analysis. Figure 7.2 compares query-by-query performance of VSM+UT (model A) and VSM+TM1- (model B) on  $\mathbf{Q}_F$ . For these analyses, I will use mean reciprocal rank (MRR)<sup>15</sup> as my metric.

On the y-axis of Figure 7.2 is the absolute difference in mean reciprocal rank between the systems for a particular query. In Figure 7.2 queries are placed on the x-axis as unique integer identifiers. A table mapping these integer identifiers to CUMTC queries is given in Table D.15 in the appendix<sup>16</sup>.

From Figure 7.2 we can observe that there in most cases, VSM+UT (blue bars) achieves a higher MRR, and that in fact there are only three queries where VSM+TM1- (orange bars) performs noticeably better than VSM+UT. In two of these the difference in reciprocal rank is close to 1.0, indicating that VSM+TM1- boosted the rank of relevant documents near the top. However, while the gains that VSM+UT produces are smaller, they are far more consistent.

If we look at documents rather than queries, a slightly different image emerges. The confusion matrix in Table 7.10, which reports performance over relevant documents (pooled from all queries), shows that there are 82 relevant documents where VSM+UT produces better ranks than VSM+TM1- (coming from 72 queries<sup>17</sup>), while VSM+TM1- produces better ranks in 36 relevant documents in total (36 queries). The models' performance is identical for 13 documents.

---

<sup>13</sup>Throughout my post-hoc analyses I will, by convention, assign model A to be the simplest of the two models and model B to be the more complex.

<sup>14</sup>Recall that there are over 439,000 individually ranked documents per run.

<sup>15</sup>The reciprocal rank is the value  $\frac{1}{r}$ , where  $r$  is the rank of a relevant document produced by a model for a particular query. Since queries can have more than one relevant document, I use mean reciprocal rank to summarise each query. For each query I compute the average reciprocal rank and plot the mean difference between models. For visualisation, reciprocal ranks have the advantage that they are positively correlated to bar size (unlike raw ranks, which are negatively correlated): better performance is represented by longer bars.

<sup>16</sup>The values on the x-axis for all bar plots of differences in mean reciprocal rank presented in the post-hoc analyses of this section map to CUMTC query IDs according to Table D.15.

<sup>17</sup>The set of queries associated with the subsets (defined over relevant documents, not queries) may overlap because some queries have more than one relevant document. This is why adding up the number of queries reported per subset does not produce the same query total as the one reported for the corpus of interest.

|          |             | VSM+TM1-   |           |             |           |
|----------|-------------|------------|-----------|-------------|-----------|
|          |             | lower rank | same rank | higher rank | not found |
| VSM + UT | lower rank  | -          | -         | 36          | -         |
|          | same rank   | -          | 13 (7)    | -           | -         |
|          | higher rank | 82         | -         | -           | 0 (1)     |
|          | not found   | -          | -         | 0 (2)       | 0 (4)     |

Table 7.10: Comparison of rankings of relevant documents: VSM+TM1- vs VSM+UT (f-t-t method, 131 relevance judgements). Boldfaced numbers in brackets refer to subset ids in Table 7.11.

For further analysis, I partitioned the corpus into the subsets shown in Table 7.11. Subsets indicate different ways how systems A and B can be better than each other (e.g. one system could outrank the other, or one system could return any relevant documents while the other doesn't). I then created the confusion matrix given in Figure 7.10, which uses the subset names from Table 7.11.

| Subset ID | Subset label                  | Description   |
|-----------|-------------------------------|---|
| (1)       | $A \setminus B$               | The corpus subset that relevant documents were returned by A but not by B.    |
| (2)       | $B \setminus A$               | The corpus subset that relevant documents were returned by B but not by A.    |
| (3)       | $A \cap B$                    | The corpus subset that relevant documents were retrieved by both models.      |
| (4)       | $\emptyset$                   | The corpus subset that relevant documents were not retrieved by either model. |
| (5)       | $A \cap B, rank(A) > rank(B)$ | The corpus subset where model A outranks B.                                   |
| (6)       | $A \cap B, rank(B) > rank(A)$ | The corpus subset where model B outranks A.                                   |
| (7)       | $A \cap B, rank(B) = rank(A)$ | Corpus subset where A and B produced the same ranks for relevant documents.   |

Table 7.11: Subsets for performance profiling pairs of models: labels and description.

I analysed the difference in average rank position between the two models for the subsets  $A \cap B, rank(A) > rank(B)$  (blue cell) and  $A \cap B, rank(B) > rank(A)$  (orange cell) and tested the difference using the permutation test. This analysis shows that in the 36 relevant documents (from 36 queries) where VSM+TM1- performed better (cell highlighted in orange), it achieved on average around 25,430 positions higher (significant at  $\alpha = 0.01$ ). In contrast, in the documents where VSM+UT performed better (cell highlighted in blue, 82 relevant documents in 72 queries), it did so on average by only 1,962 positions higher (difference significant at  $\alpha = 0.01$ ). This confirms what we have seen in Figure 7.2: the numeric advantage in MAP observed for VSM+UT is attributed to small differences in ranking in many queries.

Let us now look at the detailed properties of queries and documents in those retrieval contexts where either system has an advantage. I hypothesised that the behaviour of the models could be influenced by how many words appear in a query, how many formulae appear in it, and how complex these formulae are. Table 7.12 shows statistics about the different retrieval contexts<sup>18</sup>.

<sup>18</sup>Descriptions for the complete set of statistics I will be presenting in this section and the next can be found in

The subset identifiers in the first column of Table 7.12 correspond to the bold-faced cell labels in brackets of the confusion matrix (Table 7.10).

| Subset ID | Subset label                  | Relevant documents | Avg. No. Formulae |        | Avg. No. Words |         | Avg. Formula Complexity |       |
|-----------|-------------------------------|--------------------|-------------------|--------|----------------|---------|-------------------------|-------|
|           |                               |                    | Q                 | D      | Q              | D       | Q                       | D     |
| (0)       | $\mathbf{Q}_F$                | 131                | 9.98              | 736.13 | 117.64         | 4726.14 | 15.54                   | 24.45 |
| (3)       | $A \cap B$                    | 131                | 9.98              | 736.13 | 117.64         | 4726.14 | 15.54                   | 24.45 |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 82                 | 10.40             | 780.5  | 116.10         | 4825.18 | 15.58                   | 26.98 |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 36                 | 9.83              | 590.5  | 121.08         | 3838.97 | 15.23                   | 20.62 |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 13                 | 8.15              | 859.54 | 117.77         | 6558.15 | 14.38                   | 19.12 |

Table 7.12: Subset statistics for investigating the behaviour of VSM+UT (model A) and VSM+TM1– (model B) on  $\mathbf{Q}_F$  (tuning with f-t-t).

For each subset in Table 7.12 I list the average number of words, the average number of formulae for documents and queries (under D for relevant documents and under Q for queries). I also list the formula average complexity, which is the average number of MathML nodes per formula over the entire subset (in documents under D and queries under Q).

The latter is reported in terms of the average number of MathML nodes per formula in queries or relevant documents. I then perform significance test using the unpaired permutation test.

The document formulae in the orange subset (VSM +TM1– wins) are significantly less complex than those of the blue subset (VSM+UT wins): 20.62 vs 26.98 MathML nodes on average. I also found a marginal significant difference ( $p$ -value just above 0.05) in the average number of formulae of relevant documents (780.5 vs 590.5), with VSM+TM1– performing better on the less complex formulae. All other differences are not significant.

This means that VSM+TM1– performs better than VSM+UT when relevant documents have few words and formulae of lower average complexity; and conversely, VSM+UT performs better than VSM+TM1– when queries and documents have more words and more complex formulae. A possible explanation for this is that exact tree matching, as performed by TM1–, stumbles over slight differences between complex formulae. In contrast, UT, which is a bag-of-tuples model, is better at finding approximate matches when they exist or rejecting them when they diverge too much.

**VSM+UT vs VSM.** The most pertinent observation when comparing those two systems is that VSM+UT (model B, MAP=.071) was able to retrieve 11 relevant documents (from 11 queries) that VSM (model A, MAP=.067) was unable to find; in the methodology used before, this would correspond to (subset (2):  $B \setminus A$ )<sup>19</sup>. This is a big deal because this demonstrates that formula retrieval can have a positive effect over traditional bag-of-words models.

Beyond this achievement, VSM+UT also performs better on 71 relevant documents (from 64 queries; with a mean difference in rank of around 1, 848 positions), whereas VSM performed

Table D.3 in Appendix D.2.1.

<sup>19</sup>As already mentioned, this was obtained under f-t-t tuning, as all results in this and the next section. However, all observations reported for this posthoc analysis also hold when VSM+UT is tuned on the DevSet (cf. Table D.16, bottom).

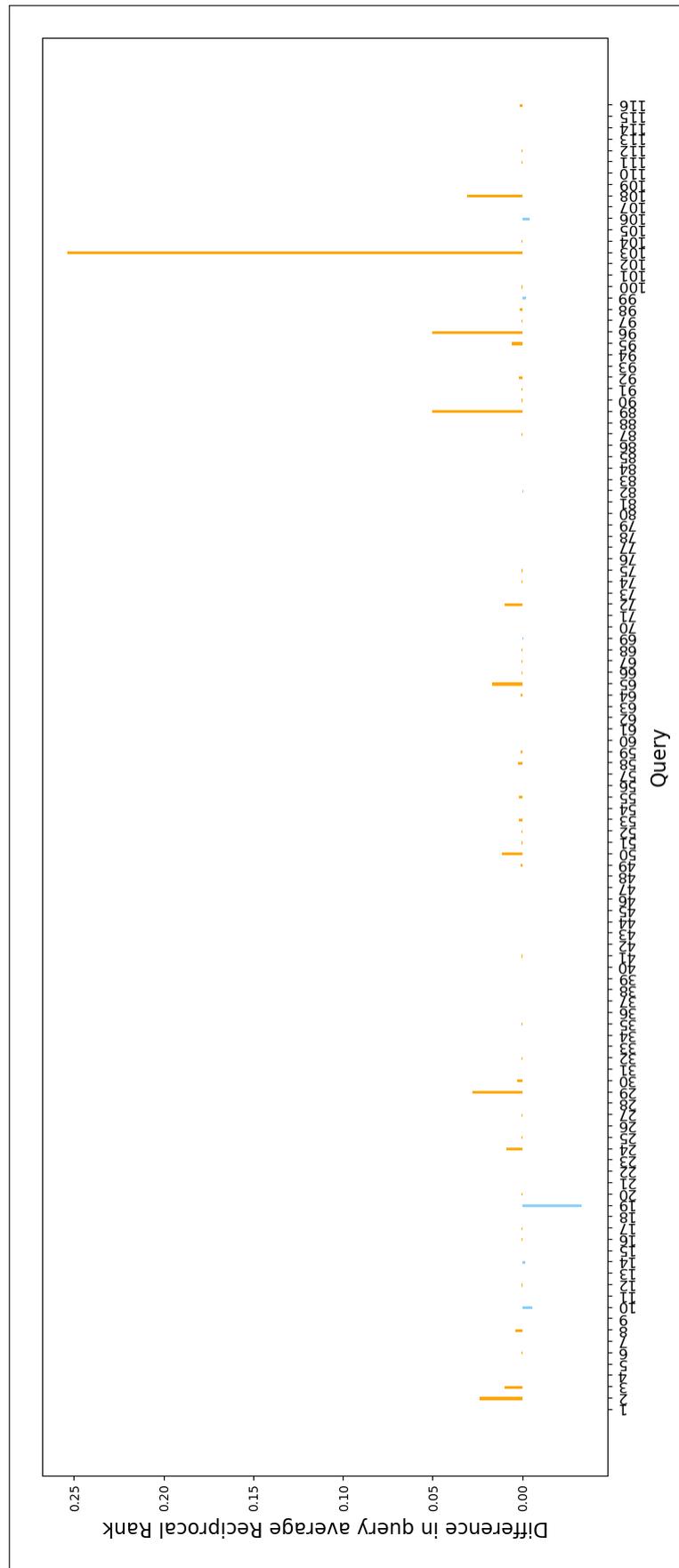


Figure 7.3: Differences in reciprocal rank between VSM (text-only baseline; blue) and VSM+UT (orange) on  $Q_F$ .

better in 30 documents (from 28 queries; with mean difference in rank of around 3,409 positions). The models were tied in their ranking in 19 relevant documents (from 19 queries).

Figure 7.3, which shows the differences in reciprocal rank between the models, drives home the fact that VSM+UT (orange) produces considerably better rankings for many queries when compared to VSM (blue). Furthermore, in the few cases VSM wins over VSM+UT, the difference in reciprocal rank is small.

I also found out about retrieval contexts where VSM+UT outperforms VSM: when there are few formulae in the queries and when the average formula complexity in either the queries or relevant documents is high.

Evidence to support this comes from the following:

1. the query average number of formulae for subset  $A \cap B, rank(B) > rank(A)$  is significantly lower than that of  $A \cap B, rank(A) > rank(B)$ : 9.61 vs 13.71;
2. the query average number of formulae for subset  $B \setminus A$  is significantly lower than that of  $A \cap B, rank(A) > rank(B)$ : 6.27 vs 13.71;
3. the average number of MathML nodes per relevant document is significantly higher in  $A \cap B, rank(B) > rank(A)$  (21,262 MathML nodes) than in  $A \cap B, rank(A) > rank(B)$  (11,369 MathML nodes), and
4. the average number of MathML nodes per formula (average formula complexity) in subset  $A \cap B, rank(B) > rank(A)$  is significantly higher than that of  $A \cap B, rank(A) > rank(B)$  (22.19 vs 27.08 nodes).

It is a little confusing that VSM+UT performs worse than VSM (i.e., if there is no formula processing at all) in the situation when there are many formulae. Why would formula processing have a negative effect? We know from the above evidence that VSM+UT performs better than VSM when the formulae in document are more complex and when there is more MathML information in them (points 3 and 4 above). So, I speculate that formula processing for VSM+UT might have a negative effect when the formulae are too simple to be informative (e.g., simple, one-letter variables having little discriminating ability). In this case, the negative effect of formula retrieval with uninformative variables pulls VSM+UT down, giving VSM an advantage.

Overall, I conclude that this negative effect is negligible in comparison to not doing any formula processing: UT's strategy of heuristically matching formula tuples gives it an overwhelming advantage over the VSM model it is based on.

#### 7.2.2.2 TM1– vs. TM2–

I now study why models based on TM2– perform worse than models based on TM1–, and I will do so with VSM baseline models and f-t-t tuning.

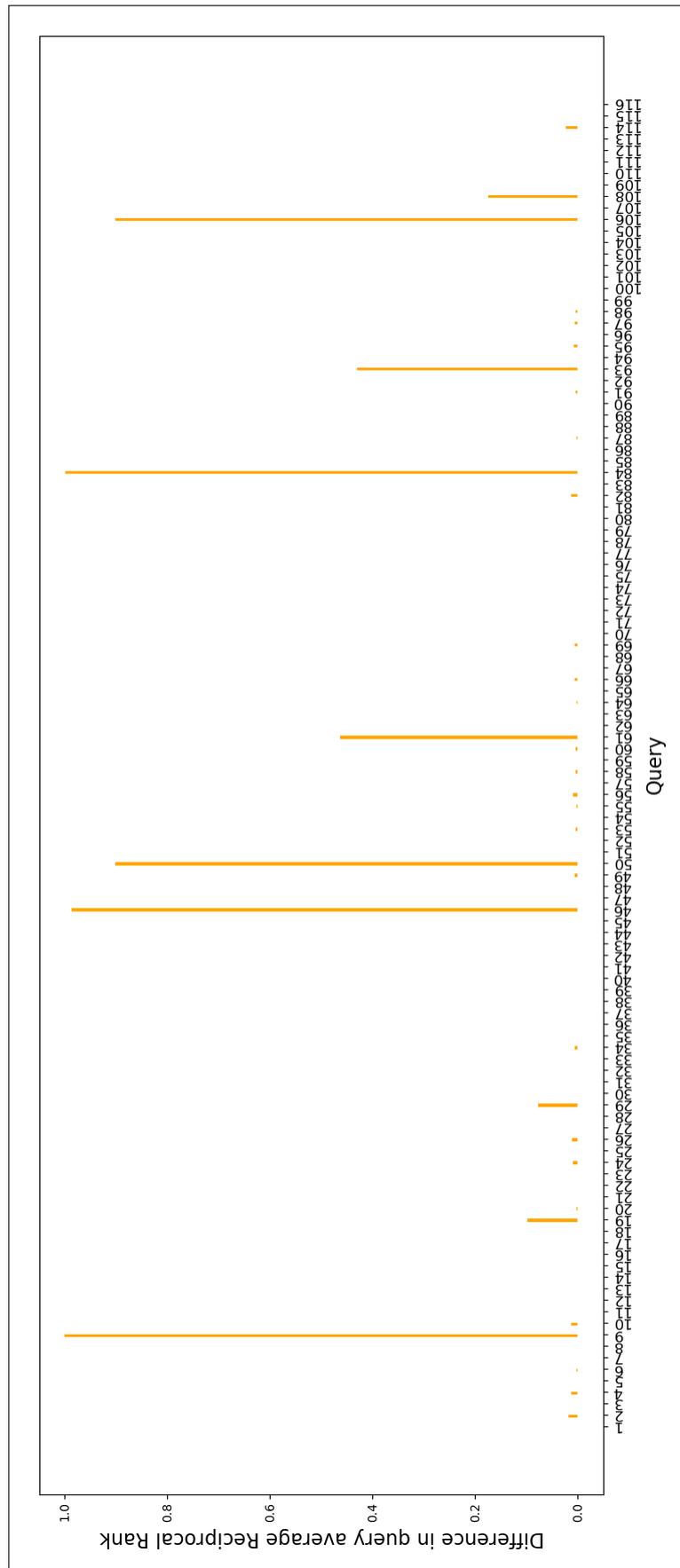


Figure 7.4: Differences in reciprocal rank between VSM+TM1- (orange) and VSM+TM2- (blue).

The mean difference in ranks on  $\mathbf{Q}_F$  (131 individually ranked documents) between the models is 441 positions, but the difference is not statistically significant.

VSM+TM2– produced higher ranks in only 18 relevant documents (from 18 queries), whereas VSM+TM1– produced higher ranks for 84 relevant documents (from 74 queries). The models produced the same rank in 29 relevant documents (from 28 queries).

Figure 7.4 shows the differences in reciprocal rank between VSM+TM1– (orange) and VSM+TM2– (blue), but the wins for VSM+TM2– are too small to be visible on the plot, supporting the observation that TM1– has consistently larger wins over TM2–.

When we consider averages over all queries, there is no big difference between how the systems ranked relevant documents: VSM+TM1– ranks relevant document on average 441 positions higher than VSM+TM2–, but the difference is not significant.

On subset (6),  $A \cap B$ ,  $rank(B) > rank(A)$ , VSM+TM1– ranked relevant documents 2,112 positions higher than VSM+TM2– on average, and this difference is also significant at level  $\alpha = 0.01$ . However, there are retrieval profiles where VSM+TM2– wins: on the 18 relevant documents of subset (5), i.e.,  $A \cap B$ ,  $rank(A) > rank(B)$ , VSM+TM2– ranked relevant documents 6,650 positions higher on average than VSM+TM1– and this difference is significant at  $\alpha = 0.01$ . From the features shown in Table 7.13 the only difference I found to be significant between subsets (5) and (6) in the table, is that (5) has fewer formulae in the queries than (6)<sup>20</sup>.

It might seem like TM2– performs best when there are fewer formulae in queries. However, this is inconclusive because it is just as likely that TM2– performed better because TM1– happened to be sensitive to the few formulae observed in those particular queries.

| Subset ID | Subset label                  | Relevant documents | Avg. No. Formulae |        | Avg. No. Words |         | Avg. Formula Complexity |       |
|-----------|-------------------------------|--------------------|-------------------|--------|----------------|---------|-------------------------|-------|
|           |                               |                    | Q                 | D      | Q              | D       | Q                       | D     |
| (0)       | $\mathbf{Q}_F$                | 131                | 9.98              | 736.13 | 117.64         | 4726.14 | 15.54                   | 24.45 |
| (3)       | $A \cap B$                    | 131                | 9.98              | 736.13 | 117.64         | 4726.14 | 15.54                   | 24.45 |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 18                 | 6.56              | 744.89 | 116.44         | 4779.06 | 14.93                   | 21.29 |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 84                 | 12.95             | 670.76 | 128.09         | 4188.36 | 14.66                   | 25.26 |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 29                 | 3.89              | 920.03 | 87.71          | 6251.0  | 17.64                   | 24.09 |

Table 7.13: Subset labels and statistics for investigating the behaviour of VSM+TM2– (model A) and VSM+TM1– (model B) on  $\mathbf{Q}_F$  (both tuned using f-t-t).

Overall, the VSM+TM1– produced numerous, substantial improvements in its ranking of relevant documents over VSM+TM2–, suggesting that the VSM+TM2– model did not benefit from the sinh transform applied to TM2.

<sup>20</sup>All other effects across subsets (5) and (6) are not significant: number of formulae (745 vs 671 formulae); number of words in the queries (18 queries in (5) and 74 queries in (6)); and number of formulae and words in relevant documents.

### 7.2.3 Comparison to Textual Models

I will now compare the performance of this section’s formula retrieval models against the textual models discussed in Section 7.1. First, Table 7.14 shows this comparison on  $\mathbf{Q}_F$  (as used throughout this section thus far).

We see that `Types2XExp` model significantly outperforms all other retrieval models we have considered so far, be they text or formula retrieval models. Also of note is the fact that `VSM+TM2-` is significantly outperformed by almost all textual models, further supporting the observation that it is the worst performing model of all models treated thus far.

|                           |            | BM25+TM1-      | BM25+TM2-      | BM25+UT        | VSM+TM1-       | VSM+TM2-       | VSM+UT         |
|---------------------------|------------|----------------|----------------|----------------|----------------|----------------|----------------|
| <b>Section 7.1 model</b>  | <b>MAP</b> | .084           | .055           | .065           | .081           | .027           | .071           |
| VSM                       | .069       | -.015          | +.014          | +.004          | -.012          | <b>**+.042</b> | -.003          |
| BM25                      | .072       | -.012          | +.017          | +.007          | -.009          | <b>**+.045</b> | +.001          |
| MLM <sub>dir</sub>        | .076       | -.008          | +.020          | +.011          | -.006          | <b>*+.048</b>  | +.004          |
| MLM <sub>dir</sub> +PRM2  | .078       | -.006          | +.023          | +.013          | -.003          | <b>*+.051</b>  | +.006          |
| MLM <sub>dir</sub> +RM3   | .090       | +.006          | +.035          | +.025          | +.009          | <b>**+.063</b> | +.019          |
| MLM <sub>jm</sub>         | .091       | +.007          | +.036          | +.026          | +.010          | <b>**+.064</b> | +.019          |
| MLM <sub>jm</sub> +RM3    | .074       | -.010          | +.018          | +.009          | -.007          | <b>*+.047</b>  | +.002          |
| MLM <sub>dir</sub> '      | .075       | -.009          | +.019          | +.010          | -.007          | <b>*+.047</b>  | +.003          |
| MLM <sub>dir</sub> ' +RM3 | .046       | -.038          | -.009          | -.019          | -.035          | +.019          | -.026          |
| SPUD                      | .098       | +.015          | +.043          | +.033          | +.017          | <b>**+.071</b> | +.027          |
| SPUD+PRM2                 | .073       | -.011          | +.018          | +.008          | -.008          | <b>*+.046</b>  | +.001          |
| SPUD+RM3                  | .044       | -.040          | -.011          | -.021          | -.037          | +.017          | -.027          |
| TF-IDFExp                 | .053       | -.030          | -.002          | -.012          | -.028          | +.026          | -.018          |
| Types2X                   | .071       | -.013          | +.016          | +.006          | -.010          | <b>*+.044</b>  | .000           |
| Types2XExp                | .156       | <b>**+.072</b> | <b>**+.101</b> | <b>**+.091</b> | <b>**+.075</b> | <b>**+.129</b> | <b>**+.084</b> |

Table 7.14: Comparison of formula retrieval models (parameters tuned with f-t-t method) against all textual retrieval models from Section 7.1 on  $\mathbf{Q}_F$  (116 queries). Significant differences in boldface (\*\* meaning  $\alpha = 0.01$  and \* meaning  $\alpha=.05$ ).

I am now changing the comparison corpus to  $\mathbf{Q}$ . This is arguably kinder to the textual models, as even if a query does not have a formula in it, the text-based models can still perform type-based processing, whereas the models in the current section cannot run at their full potential on such queries. Table 7.15 shows the comparison between the formula retrieval models and the full textual models. As before, the `Types2XExp` model remains the strongest and significantly outperforms all formula-aware models on the full test collection. On  $\mathbf{Q}$ , `VSM+TM2-` is still numerically worse than every textual model, but the difference is not as pronounced as it is on  $\mathbf{Q}_F$ .

|                          |            | BM25+TM1-      | BM25+TM2-      | BM25+UT        | VSM+TM1-       | VSM+TM2-       | VSM+UT         |
|--------------------------|------------|----------------|----------------|----------------|----------------|----------------|----------------|
| <b>Section 7.1 model</b> | <b>MAP</b> | .085           | .064           | .071           | .086           | .046           | .079           |
| VSM                      | .076       | -.009          | +.012          | +.005          | -.010          | <b>*+.030</b>  | -.003          |
| BM25                     | .077       | -.008          | +.013          | +.006          | -.009          | <b>*+.031</b>  | -.002          |
| MLM <sub>dir</sub>       | .066       | -.019          | +.002          | -.006          | -.020          | +.019          | -.013          |
| MLM <sub>dir</sub> +PRM2 | .061       | -.024          | -.002          | -.010          | -.024          | +.015          | -.017          |
| MLM <sub>dir</sub> +RM3  | .082       | -.003          | +.018          | +.010          | -.004          | +.035          | +.003          |
| MLM <sub>jm</sub>        | .084       | -.001          | +.020          | +.013          | -.001          | +.038          | +.005          |
| MLM <sub>jm</sub> +RM3   | .063       | -.022          | -.001          | -.009          | -.023          | +.016          | -.016          |
| MLM' <sub>dir</sub>      | .072       | -.013          | +.009          | +.001          | -.013          | +.026          | -.006          |
| MLM' <sub>dir</sub> +RM3 | .051       | -.034          | -.013          | -.021          | -.035          | +.004          | -.028          |
| SPUD                     | .090       | +.005          | +.027          | +.019          | +.005          | <b>*+.044</b>  | +.012          |
| SPUD+PRM2                | .072       | -.014          | +.008          | .000           | -.014          | +.025          | -.007          |
| SPUD+RM3                 | .050       | -.035          | -.014          | -.021          | -.036          | +.004          | -.029          |
| TF-IDFExp                | .060       | -.025          | -.003          | -.011          | -.025          | +.014          | -.018          |
| Types2X                  | .070       | -.015          | +.006          | -.002          | -.016          | +.023          | -.009          |
| Types2XExp               | .173       | <b>**+.088</b> | <b>**+.109</b> | <b>**+.101</b> | <b>**+.087</b> | <b>**+.126</b> | <b>**+.094</b> |

Table 7.15: Comparison of formula retrieval models (parameters tuned on f-t-t) with textual retrieval models from Section 7.1 on **Q** (160 queries).

For completeness, the appendix contains the comparison tables for  $Q_{FT}$  (Tables D.13 and D.14 for f-t-t and Devset methods, respectively) and the tables corresponding to Table 7.14 and 7.15 when  $\alpha$  is tuned using the DevSet (Tables D.5 and D.9, respectively).

### 7.3 Joint Retrieval: Typed Text and Typed Formulae

I will now turn to my experiments with joint typed retrieval models, those that utilise typed SLTs as the connection of textual and symbolic modalities. The models in the current section, unlike those in Section 7.2, have full access to type information (both textual and formula components), which means that the tree models can operate with type unification. Using these typed models I can now investigate my second hypothesis, namely: retrieval models that use types extracted from the text as denotations to symbols in formulae will have better retrieval efficiency, compared to those models than those that do not utilise this link between textual and symbolic modalities (i.e., that treat text and formulae independently).

#### 7.3.1 Experimental Design

Joint retrieval models have a textual component and a formula component. The textual components are repeated for the reader's convenience in Table 7.16. To remind the reader, in joint models, formulae are removed from the textual components; the textual components operate only on words, types or both. The scoring of formulae is instead delegated to the type-aware formula scoring components. Formula and textual scores are then combined using linear interpolation, as described in Section 7.2.

The no-formula textual components in Table 7.16 can be considered as a baseline that joint retrieval models should be able to easily outperform. Practically, these no-formula baselines (reported as “none” in results tables in this section) are created by setting  $\alpha=1$ . In all other cases, the results presented in this section estimate  $\alpha$  by f-t-t estimators<sup>21</sup>.

| <b>Textual Component</b> | <b>Description</b>  |
|--------------------------|---|
| Type2XExp                | The textual context score is obtained using the Types2XExp model: Lucene VSM with type-based query expansion first introduced in section 7.1.                                     |
| Types2X                  | The textual context score is obtained using the Types2X model (Lucene VSM with types re-written in the query and documents and given a 2X boost) first introduced in section 7.1. |
| Types                    | Scores for the textual context are obtained by replacing type phrases with atomic lexical tokens and querying a Lucene index with types also replaced with atomic lexical tokens. |
| VSM                      | Scores for the textual context are obtained using of-the-box Lucene bag-of-words retrieval with no formulae in the queries.   |
| BM25                     | Scores for the textual context are obtained using out-of-the-box BM25 bag-of-words retrieval with no formulae in the queries.   |

Table 7.16: Textual, type-aware baselines used in typed retrieval.

Table 7.17 lists the type-aware formula models considered in this section.

<sup>21</sup>Based on the results from from section 7.2, f-t-t is assumed to be the overall better tuning method; I have nevertheless also used the devset estimator, and appendix D.3.2.1 shows results from DevSet tuning which is indeed overall less beneficial for all systems and also produces a lower winner.

| Formula component | Description   |
|-------------------|---|
| TM1               | Tree matching model 1 with type-based unification, i.e., the semantic component that measures the number of typed nodes that either match directly or can be matched via type expansion or suffix tree traversal ( $semantic_{(QF_i, DF_j)}$ sub-score, as discussed in Section 6.4.1). |
| TM2               | Tree matching model 2 with type-based unification.  |
| TM1 <sub>JD</sub> | Tree matching model 1 with “semantic overlap” between query and document formulae measured using the Jaccard coefficient over types ( $semantic_{(QF_i, DF_j)_{JD}}$ sub-score, as discussed in Section 6.4.1).   |
| TM2 <sub>JD</sub> | Tree matching model 2 with “semantic overlap”.  |
| TM1 <sub>JE</sub> | Tree matching model 1 with “expanded Jaccard semantic overlap” between query and document formulae measured using the Jaccard coefficient over expanded query types ( $semantic_{(QF_i, DF_j)_{JE}}$ sub-score, as discussed in Section 6.4.1).   |
| TM2 <sub>JE</sub> | Tree matching model 2 with “expanded Jaccard semantic overlap” over expanded types.   |
| TT                | My own typed variant of Tangent, as discussed in Section 6.4.3.   |

Table 7.17: Type-aware symbolic retrieval components used in typed retrieval.

I introduced semantic overlap in Section 6.4.2 as a simpler, less strict alternative to type unification for semantic scoring of formulae based on types. I will briefly recapitulate the idea here: semantic overlap for two formulae, one coming from the document and the other from the query, is calculated as follows. First the types associated to the formulae by variable typing and type disambiguation are collected into two sets. Then, the formula pair is scored by computing the Jaccard coefficient of their type sets ( $semantic_{(QF_i, DF_j)_{JD}}$  sub-score, as discussed in Section 6.4.1). Here, the Jaccard coefficient models the degree to which the types used in the document and query formulae overlap in a structurally independent manner (i.e., the structure of the formulae is ignored).

Variants of my tree matching models that use the JE method take this a step further by also expanding the set of types for the query formula using the type embedding space (Section 4.3). Overlap in the JE method is calculated over the expanded sets of types to model more in-depth topical similarity between two formulae.

As described in Section 6.4.1, type unification can be substituted with semantic overlap by replacing the semantic scoring component of my tree matching models, i.e., replace  $semantic_{(QF_i, DF_j)}$  with  $semantic_{(QF_i, DF_j)_{JD}}$  or  $semantic_{(QF_i, DF_j)_{JE}}$ . This substitution produces the models TM1<sub>JD</sub>, TM1<sub>JE</sub>, TM2<sub>JD</sub> and TM2<sub>JE</sub>.

My experiments with typed retrieval are centered around  $\mathbf{Q}_{FT}$  because this subset of my test collection contains valid variable typings with at least one typable formula in every query.

Therefore,  $Q_{FT}$  simulates conditions where typed retrieval can be effectively evaluated. Again, and strictly for reasons of comparison with models from sections 7.1 and 7.2, I will also report performance on  $Q$  and  $Q_F$ .

### 7.3.2 Results and post-hoc analyses

Table 7.18 shows the performance of all formula and text retrieval models, including joint retrieval models, on  $Q_{FT}$ , with the f-t-t tuning method. The highest performing model,  $TM1_{JE}$  achieves  $MAP=.151$ . This table reports whether the systems manage to significantly improve over their respective baselines; boldfaced differences are significant. We can see that this is the case for systems  $VSM+UT$ ,  $VSM+TT$ ,  $VSM+TM2_{JE}$ ,  $Types+TM2-$ ,  $Types2XExp+TM2-$ ,  $Types2XExp+UT$ ,  $Types2XExp+TM2_{JD}$  and  $Types2XExp+TM1_{JE}$ .

| Textual model (baseline) |              | None | Formula Model |              |              |              |        |        |                    |                    |                    |                    |
|--------------------------|--------------|------|---------------|--------------|--------------|--------------|--------|--------|--------------------|--------------------|--------------------|--------------------|
|                          |              |      | Untyped       |              |              | Typed        |        |        |                    |                    |                    |                    |
|                          |              |      | +TM1-         | +TM2-        | +UT          | +TT          | +TM1   | +TM2   | +TM1 <sub>JD</sub> | +TM2 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2 <sub>JE</sub> |
| VSM                      | MAP          | .065 | .082          | .026         | .070         | .069         | .075   | .074   | .084               | .036               | .082               | .039               |
|                          | MAP $\Delta$ |      | +0.017        | <b>-.039</b> | <b>+.005</b> | <b>+.004</b> | +0.010 | +0.009 | +0.019             | -.029              | +0.017             | <b>-.026</b>       |
| BM25                     | MAP          | .067 | .087          | .057         | .065         | .063         | .085   | .084   | .084               | .064               | .085               | .065               |
|                          | MAP $\Delta$ |      | +0.020        | -.010        | -.002        | -.004        | +0.018 | +0.017 | +0.017             | -.003              | +0.018             | -.002              |
| Types                    | MAP          | .039 | .053          | .028         | .039         | .040         | .039   | .047   | .050               | .033               | .050               | .034               |
|                          | MAP $\Delta$ |      | +0.014        | <b>-.011</b> | .000         | +0.001       | .000   | +0.008 | +0.011             | <b>-.006</b>       | +0.011             | -.005              |
| Types2X                  | MAP          | .068 | .089          | .044         | .062         | .061         | .076   | .083   | .077               | .051               | .082               | .061               |
|                          | MAP $\Delta$ |      | +0.021        | -.024        | -.006        | -.007        | +0.008 | +0.015 | +0.009             | -.017              | +0.014             | -.007              |
| Types2XExp               | MAP          | .131 | .147          | .069         | .133         | .123         | .140   | .142   | .150               | .096               | .151               | .106               |
|                          | MAP $\Delta$ |      | +0.016        | <b>-.062</b> | <b>+.002</b> | -.008        | +0.009 | +0.011 | +0.019             | <b>-.035</b>       | +0.020             | <b>-.025</b>       |

Table 7.18: Performance of untyped and typed (joint) formula retrieval models on  $Q_{FT}$  (106 queries), including differences from respective text-only baseline model (bold if significant).

Overall, it is clear that systems not based on  $Types2XExp$  are not competitive, so I will focus my further analyses on models based on  $Types2XExp$  (bottom lowest row). I will henceforth abbreviate models based on  $Types2XExp$  by only referring to their formula component. For example,  $+TM1$  abbreviates  $Types2XExp+TM1$  and so forth.

If we look at the  $Types2XExp$ -based models, we see a surprising effect: although several models perform numerically well above baseline (e.g.,  $+TM1-$  (.147),  $+TM1$  (.140),  $+TM1_{JE}$  (.151; numerically the highest performance),  $+TM1_{JD}$  (.150), and  $+TM2$  (.142)), these differences are not significant, whereas  $UT$ 's performance, which is only 0.133, *is* significantly different from its baseline at .131, despite the small effect size. We have noticed a similar effect earlier. This is part of the reason why I will perform some post-hoc analyses to investigate this further.

We also notice again a marked difference between tree models  $TM1$  and  $TM2$ . Earlier, we have seen  $TM2-$  to be far inferior to  $TM1-$  (and seen the corresponding post-hoc analysis). Here, we see that  $TM2$  performs relatively well when using type unification (in its  $TM2+$  incarnation), but in all other configurations (untyped or with semantic component  $JD$  or  $JE$ ), it falls far below  $+TM1$ 's respective performance. In fact, the three systems  $+TM2-$ ,  $+TM2_{JD}$ ,  $TM2_{JE}$  are the only  $Types2XExp$ -based systems which performed numerically below their no-formula baseline

(here: MAP=.131). I conclude that while TM2 cannot benefit from sophisticated semantic similarity metrics such as JD and JE and actually performs numerically worse, TM1 can profit from these. TM1 benefits numerically from typed unification (TM1) as well as from the JD and JE methods, but TM2 only benefits from typed unification (TM2).

|                    |            |      |      |      |      |                    |                    |
|--------------------|------------|------|------|------|------|--------------------|--------------------|
| +TT                | .123       | ≪    |      |      |      |                    |                    |
| +TM1               | .140       | ≈    | ≈    |      |      |                    |                    |
| +TM2               | .142       | ≈    | ≈    | ≈    |      |                    |                    |
| +TM1 <sub>JD</sub> | .150       | ≈    | ≈    | ≈    | ≈    |                    |                    |
| +TM1 <sub>JE</sub> | .151       | ≈    | ≈    | ≈    | ≈    | ≈                  |                    |
| +TM1-              | .147       | ≈    | ≈    | ≈    | ≈    | ≪                  | ≪                  |
|                    | <b>MAP</b> | .133 | .123 | .140 | .142 | .150               | .151               |
|                    |            | +UT  | +TT  | +TM1 | +TM2 | +TM1 <sub>JD</sub> | +TM1 <sub>JE</sub> |

Table 7.19: Significance results between selected untyped and typed Types2XExp-based retrieval models on  $Q_{FT}$  ( $\alpha$  tuned using f-t-t).

Table 7.19 gives the significance of differences between 7 seemingly competitive typed and untyped retrieval models based on Types2XExp (i.e., those from Table 7.18 that numerically perform above baseline), namely the two untyped models TM1- (.147) and UT (.133), and the typed models TM1 (.140), TM1<sub>JE</sub> (.151), TM1<sub>JD</sub> (.150), and TM2 (.142). I also added typed Tangent (TT) into this mix because I want to know how it compares to other typed and untyped formula scoring models.

Unfortunately, it is impossible to establish any differences between these systems, except the following: First, +UT (.133) is significantly better than +TT (.123)<sup>22</sup>; second, +TM1<sub>JD</sub> (.150 MAP) and +TM1<sub>JE</sub> (.151 MAP) significantly outperform +TM1- (.147 MAP). Note that in the case of the pairs +TM1- vs +TM1<sub>JD</sub> and +TM1- vs +TM1<sub>JE</sub> this significance is achieved despite the very small effect size (.003 and .004, respectively).

The first observation is surprising because both +UT and TT use the same structural heuristic, with the latter also indexing types in the Tangent tuple table.

The second observation suggests that TM1 benefits from the addition of a semantic similarity based on types, but not from type unification per se (the difference between +TM1 and +TM1- is not significant according to Table 7.19); it is only in combination with semantic overlap (JD and JE) that +TM1 manages to show significant improvements over +TM1-. When combined with type unification, it is statistically indistinguishable from +TM1-.

Another, less relevant significant difference is that all high-performers are significantly better than the three low-performers (TM2<sub>JE</sub>, TM2<sub>JD</sub> and the no-type unification TM2-)<sup>23</sup>, with +TM2- being the weakest model: it is significantly outperformed by all other models, an observation that is consistent with the results in section 7.2.

<sup>22</sup>Confusingly, although all 5 TM-based high-performers are numerically better than UT, there is no significant difference between either of them and TT, another instance of the aforementioned anomaly of statistical testing.

<sup>23</sup>For brevity, the results of the significance tests on these systems are not shown here. See Table D.3.1 in Appendix D for the full significance table.

All analyses up to this point in this section were performed on  $Q_{FT}$ , the corpus which is most informative for typed retrieval. For completeness and comparability, appendix D.3.2 lists all typed retrieval results on all corpora ( $Q_F$ ,  $Q$  and DevSet results for  $Q_{FT}$ ).

I will now perform three post-hoc analyses in an attempt to shine light on three additional questions:

1. I built the machinery for typed unification in Chapter 6, which is embodied by the tree matching models with typed unification (TM1 and TM2 formula components). I was surprised to see that these models did not perform consistently and significantly better than their no-unification counter-parts (TM1– and TM2–). I want to know why this is the case, and investigate by comparing TM1 with TM1–.
2. In the next set of analyses, I study why semantic overlap, in the form of +TM1<sub>JE</sub>, performs as well as it does. I look at two situations: why does it perform so well when compared to tree matching with no types (i.e., +TM1–)? And why does it perform *worse* than if no formula matching was ever performed (examined by comparing it with full Types2XExp from Section 7.1, the best-performing system, which uses only textual types)<sup>24</sup>?
3. How does the best performing joint typed retrieval model, Types2XExp+TM1<sub>JE</sub>, compare to standard Tangent (Pattaniyil and Zanibbi, 2014), VSM+UT? This addresses the question of how one of the best currently existing out-of-the box MathIR models compare to the best of the typed retrieval systems invented in this thesis or combined in a novel manner from existing parts.

To answer the questions above, I will re-apply the analysis methods from Sections 7.1 and 7.2 on  $Q_{FT}$ , which is the corpus of interest. I also for the first time add statistics about query and document types and typings to the subset statistics from previous sections, as earlier systems were untyped (cf. Table D.3 for a description of all subset statistics).

### 7.3.2.1 Performance of type unification

I investigate the first question by performing post-hoc analyses on

- +TM1 (.140) vs +UT (.133);
- +TM1 (.140) vs +TM1– (.147); and
- +TM1 (.140) vs +TM1<sub>JE</sub> (.151).

<sup>24</sup>At the danger of repeating myself, this is different from asking whether Types2XExp+TM1<sub>JE</sub> beats its textual Types2XExp baseline (we have found out in the current section that it does). The difference between the textual Types2XExp baseline and the full Types2XExp system from section 7.1 lies in how formulae are treated: in 7.1, formulae are indexed as text strings, whereas in the baseline systems in 7.3, formulae are dropped entirely for the baseline (and treated by a dedicated formula component for the full joint systems). The 7.1 version can therefore be expected to be much stronger.

I performed these comparisons despite neither of these pairs being significant different when measured in MAP (cf. Table 7.19). Note that I will only discuss in detail those pairs that exhibit distinct behaviour with mentions to common patterns.

Here, +TM1 represents type unification, while the other models represent the best alternatives: +UT for heuristic-based raw symbol formula retrieval, +TM1– for no-unification tree matching, and +TM1<sub>JE</sub> for semantic overlap.

**+TM1 vs +UT:** Figure 7.5<sup>25</sup> compares +TM1 to +UT, in terms of differences in mean reciprocal rank across the queries. We observe that +UT (blue) produces slightly better ranks than +TM1 for many queries. In the opposite case, where +TM1 (orange) produced better ranks than +UT, it does so by ranking very few relevant documents considerably higher than +UT.

This behaviour could explain why the difference between +UT and +TM1 is not significant: +TM1’s MAP improvements over +UT are attributable to a very small number of well-performing queries. I will now look for significant differences between the specific subsets of the corpus where A (+UT) wins over B (+TM1) (subset (5)) and vice-versa (subset (6)).

Statistics for these subsets are shown in Table 7.20. Note that in Table 7.20 the statistic “Avg Total MathML Nodes” is the mean number of MathML nodes of the subset (queries or relevant documents). +UT produces better ranks than +TM1 in 68 relevant documents (from 62 queries) while model B performs better in 32 relevant documents (from 31 queries).

Significant differences across the subsets can be found in the following aspects: number of words (5106.96 vs 3609.66), number of formulae (802.76 vs 536.56), total MathML nodes (21,154.31 vs 12,113.69) and average formula complexity (26.62 vs 20.65) in relevant documents (the corresponding cells are highlighted in Table 7.20). In other words, +TM1 tends to perform better than +UT when there are fewer words and formulae in the relevant documents, and when the formulae are of lower complexity, but +UT seems to be able to take better advantage of all of this information (more text, more formulae and formulae of higher complexity) in relevant documents.

Overall, +UT’s performance is more consistent than that of +TM1. However, +TM1 may be advantageous in retrieval scenarios where recall is a priority<sup>26</sup>. In these scenarios, the many, small losses in precision by TM1 represented by the blue differences (cases where UT wins) might not be so important to searchers, as they accept overall low precision anyway. However, the few cases where +TM1 ranks relevant documents much higher than +UT might be a large advantage to them, as they don’t have to go down the retrieval rankings a lot. This, in my interpretation, makes +TM1’s behaviour at least as acceptable as +UT’s in real-life MIR because

---

<sup>25</sup>Queries on the x-axis of all bar-plots of differences in mean reciprocal rank in this section are again mapped onto integer identifiers. Table D.28 in the appendix can be used to map the integer identifiers back to CUMTC query IDs for all bar plots in this section.

<sup>26</sup>This happens to be the case in my corpus, where there are few relevant documents per query and where searchers might therefore be motivated to accept lower precision because they are happy to inspect more documents in order to get higher recall.

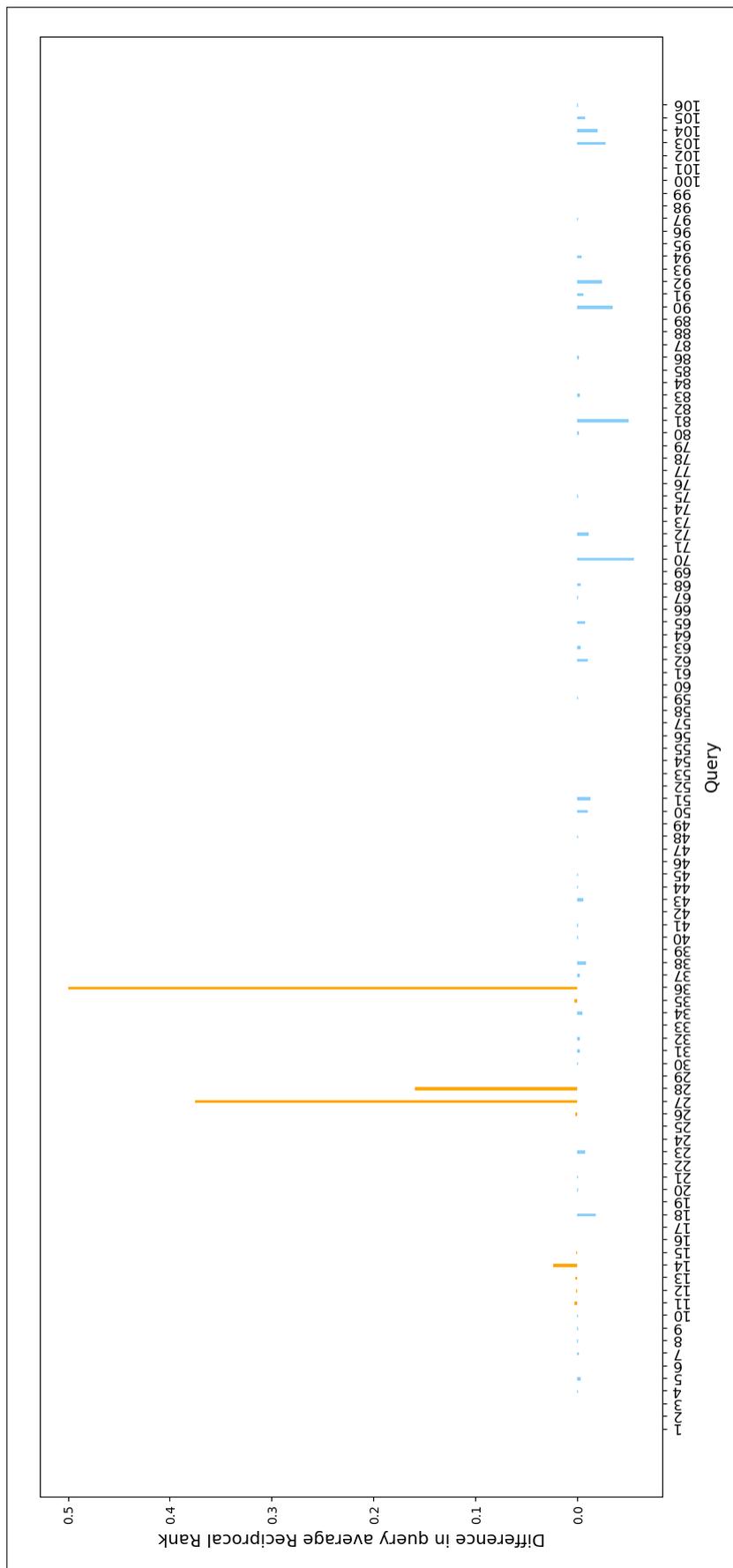


Figure 7.5: Differences in reciprocal rank between +TM1 (orange) and +UT (blue) on  $Q_{FT}$ .

| Subset ID | Subset label                  | Relevant documents | Avg. No. Formulae         |              | Avg. No. Words            |              |
|-----------|-------------------------------|--------------------|---------------------------|--------------|---------------------------|--------------|
|           |                               |                    | Q                         | D            | Q                         | D            |
| (0)       | $\mathbf{Q}_{FT}$             | 118                | 10.63                     | 718.53       | 124.17                    | 4669.71      |
| (3)       | $A \cap B$                    | 118                | 10.63                     | 718.53       | 124.17                    | 4669.71      |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 68                 | 10.69                     | 802.76       | 120.52                    | 5106.69      |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 32                 | 11.71                     | 536.56       | 120.87                    | 3609.66      |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 18                 | 8.47                      | 723.83       | 141.29                    | 4903.44      |
| Subset ID | Subset label                  | Relevant documents | Avg. No. Types            |              | Avg. No. Positive Typings |              |
|           |                               |                    | Q                         | D            | Q                         | D            |
| (0)       | $\mathbf{Q}_{FT}$             | 118                | 17.83                     | 879.8        | 2.77                      | 12817505.56  |
| (3)       | $A \cap B$                    | 118                | 17.83                     | 879.8        | 2.77                      | 12817505.56  |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 68                 | 18.08                     | 953.18       | 2.73                      | 13563403.24  |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 32                 | 16.35                     | 681.59       | 3.42                      | 12102616.31  |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 18                 | 19.12                     | 954.94       | 1.94                      | 11270584.11  |
| Subset ID | Subset label                  | Relevant documents | Avg. No. Negative typings |              | Avg. No. Typing Edges     |              |
|           |                               |                    | Q                         | D            | Q                         | D            |
| (0)       | $\mathbf{Q}_{FT}$             | 118                | 19.32                     | 135966583.4  | 22.09                     | 148784088.96 |
| (3)       | $A \cap B$                    | 118                | 19.32                     | 135966583.4  | 22.09                     | 148784088.96 |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 68                 | 20.48                     | 143891909.47 | 23.21                     | 157455312.71 |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 32                 | 18.39                     | 128344728.31 | 21.81                     | 140447344.63 |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 18                 | 14.88                     | 119576427.28 | 16.82                     | 130847011.39 |
| Subset ID | Subset label                  | Relevant documents | Avg. total MathML Nodes   |              | Avg. Formula Complexity   |              |
|           |                               |                    | Q                         | D            | Q                         | D            |
| (0)       | $\mathbf{Q}_{FT}$             | 118                | 151.02                    | 17703.15     | 14.89                     | 24.11        |
| (3)       | $A \cap B$                    | 118                | 151.02                    | 17703.15     | 14.89                     | 24.11        |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 68                 | 159.06                    | 21154.31     | 14.44                     | 26.62        |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 32                 | 157.68                    | 12113.69     | 16.57                     | 20.65        |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 18                 | 106.94                    | 14602.28     | 13.01                     | 20.76        |

Table 7.20: Per-query difference in average reciprocal rank between +UT (blue) and +TM1 (orange).

+TM1 is more likely to bring these difficult to find relevant documents to the attention of the searcher.

However, this phenomenon does not occur often enough in my test collection to be consistent and to produce a detectable and significant positive effect over +UT<sup>27</sup>.

**+TM1 vs +TM1-** I now turn my attention to the comparison of +TM1 (model B) with +TM1- (model A). I found that +TM1- produces better ranks than its unification counterpart +TM1 in 29 relevant documents (from 28 queries) while +TM1 produced better ranks than +TM1- in 63 relevant documents (from 59 queries). Figure 7.6 shows the differences in mean reciprocal rank between the models on  $\mathbf{Q}_{FT}$  (blue for +TM1- and orange for +TM1).

From Figure 7.6 we observe that when +TM1 wins it does so by producing slightly better ranks for many queries (with one exception where the difference is relatively high). However, +TM1- produced considerably better ranks for three queries, which may be the main contributors to +TM1-'s numerically (but not significantly) better performance<sup>28</sup>.

<sup>27</sup>I would like to remind the reader that my observations are conditioned on the tuning method.

<sup>28</sup>As to the circumstances when +TM1 performs better than +TM1-, I examined again the differences in the various statistics across subsets (5) and (6), but found no significance except that in average formula complexity (21.34 vs 26.53 MathML nodes per formula) in relevant documents, corresponding to a slight tendency of +TM1 to perform better than +TM1- when the formulae in documents are relatively large in terms of average MathML nodes.

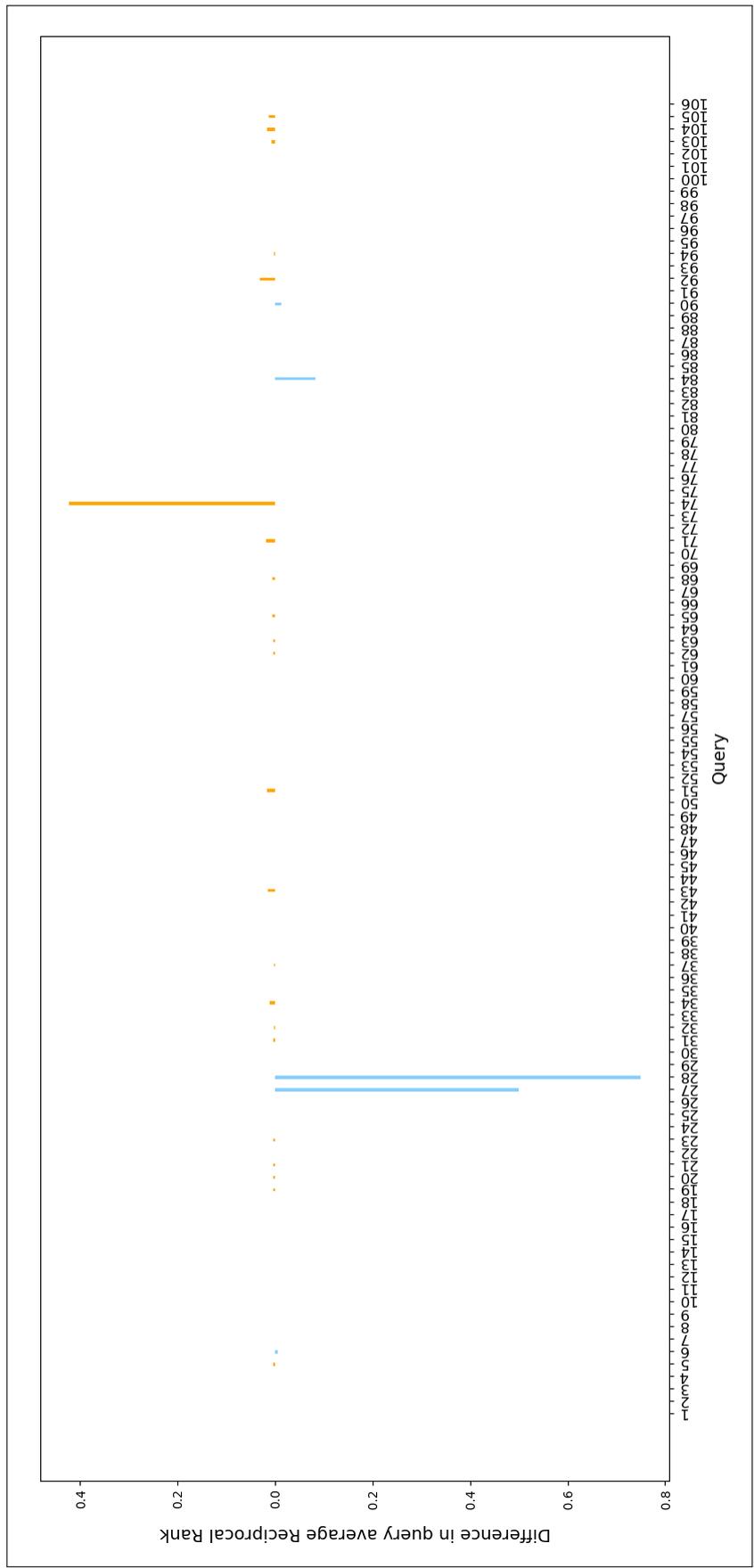


Figure 7.6: Per-query difference in average reciprocal rank between +TM1- (blue) and +TM1 (orange) on  $Q_{FT}$ .

**+TM1 vs +TM1<sub>JE</sub>:** When comparing +TM1 (model A) to +TM1<sub>JE</sub> (model B), I found the same pattern in ranking behaviour observed in the pair +TM1– vs +TM1: model B produced smaller but more numerous improvements (one query with larger improvement than the majority of queries) but model A produced large improvements in rank for three queries. Looking into the differences between subsets (5) and (6) for this pair, I found differences in the average total number of MathML nodes (20,784.11 vs 12,895.13) and average formula complexity (27.94 vs 19.95) in relevant documents to be statistically significant. This is in line with what we saw earlier: +TM1 performs best when the formulae in relevant documents are relatively large or complex.

We have seen that in comparison to my other tree matching models +TM1 produces numerous small improvements but is numerically outperformed by a few queries where the improvement is sizeable against it. This pattern is opposite of that observed when comparing +TM1 against +UT.

Overall, I have learned the following from comparing the three pairs:

1. TM1 can rank a small number of relevant documents from a few queries substantially higher than +UT. Gains by +UT over +TM1 are numerous but relatively small, where as gains by +TM1 over +UT can be substantial in a small number of cases.
2. The fact that +TM1 results in MAP regression over +TM1– and +TM1<sub>JE</sub> suggests that type unification does not yield consistent, positive effects, but this is inconclusive since the difference between the models is not significant.
3. My tree matching models are, in general, capable of performing very well in a small number of queries.
4. Of all the tree matching models examined in this analysis, +TM1 is the riskiest as it can produce large, positive spikes in rank, but that these occurrences are rare and come at the cost of many losses (which can be sizeable when compared to TM1– and TM1<sub>JE</sub>).

### 7.3.2.2 Performance of semantic component

I will now change focus of analysis to the overall best joint model, +TM1<sub>JE</sub>, as I want to analyse in which way this tree model's semantic component achieves its relatively good performance. We have seen in the last section that +TM1<sub>JE</sub>'s strategy of swapping typed unification with semantic overlap is overall beneficial. Here I will compare it to +TM1–, which does tree matching without the use of types (i.e., it does not incorporate any semantic component in its formula pair similarity function).

**+TM1<sub>JE</sub> vs. +TM1–:** The difference in MAP between +TM1– (model A, MAP=.147) and +TM1<sub>JE</sub> (model B, MAP=.151) is small (.004) but significant at  $\alpha = 0.01$ . When I apply the same methodology as before, I find that model B produces better ranks (for 69 relevant

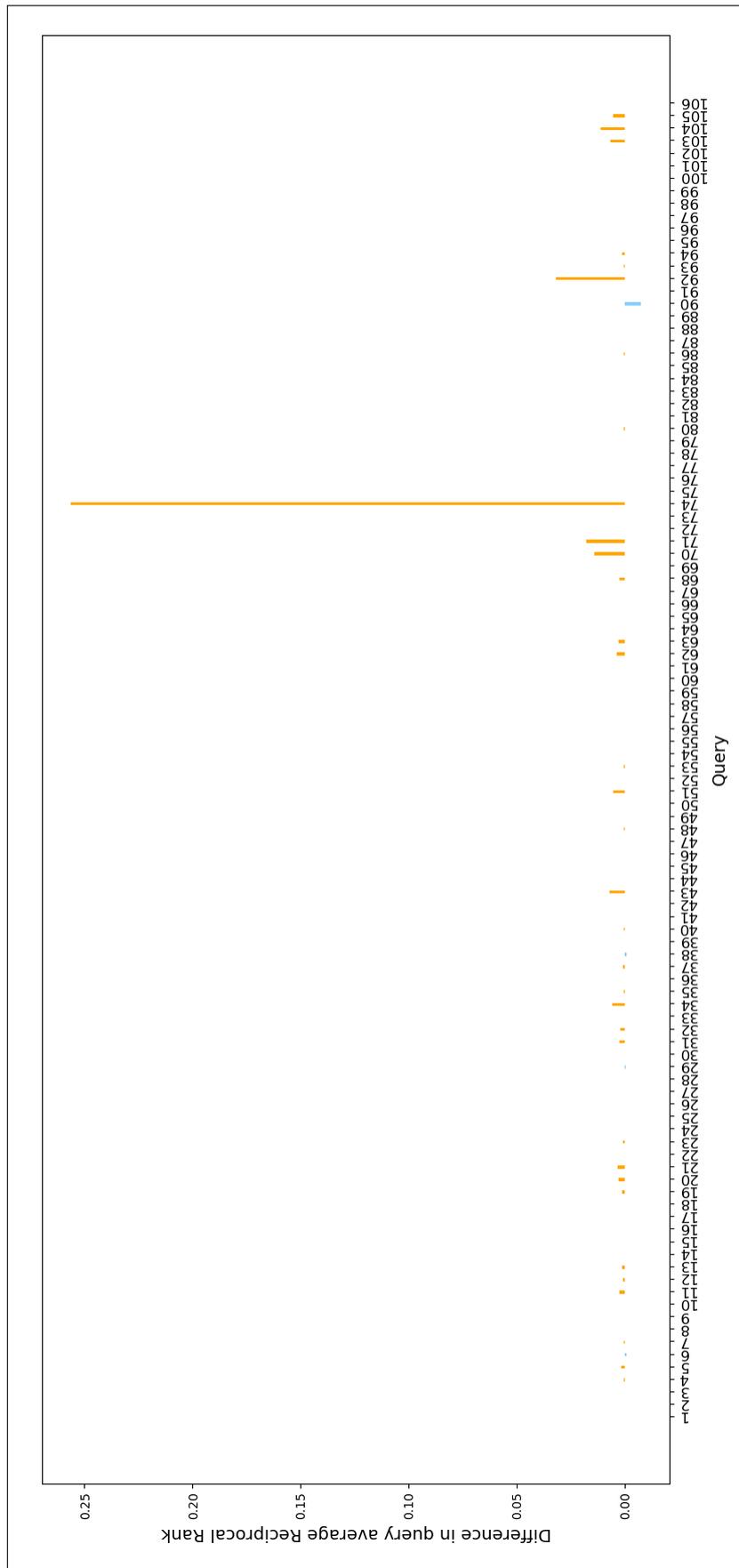


Figure 7.7: Per-query difference in average reciprocal rank between Types2XExp+TM1- (blue) and Types2XExp+TM1<sub>JE</sub> (orange) on  $\mathbf{Q}_{FT}$ .

documents (from 64 queries; subset (6)), whereas model A produces better ranks for 20 relevant documents (from 19 queries; subset (5))<sup>29</sup>.

Turning to a per-query view, Figure 7.7 illustrates that when model B (orange bars) wins over model A (blue bars) it appears to do so more consistently. Specifically, looking at the orange bars in Figure 7.7 we see that model B produces very small improvements in reciprocal rank for many queries, with only a single query making a sizeable improvement. In contrast, when differences in reciprocal rank are in favour of model A (blue bars), they are too small to be visible in the figure.

These observations suggest that adding the *JE* semantic overlap component to TM1– can have a consistent, positive effect. We also find here likely explanations as to why the difference is significant, despite the small effect size of  $MAP=.004$ .

Taking the results of all post-hoc analyses discussed so far into consideration, there are two conclusion I can draw. First, +TM1<sub>JE</sub> is the least risky tree matching model: it can produce consistent improvements in rank over +TM1–, however small these differences may be. Second, tree matching (in general) has the potential to work really well. However, there are too few cases where this is observed (in my test collection, at least) and too many cases where the effect is slightly negative, affecting the overall result. Nevertheless, this is potentially a hopeful result, because one can maybe develop a scoring architecture and models where the small errors can be reduced while being offset by few queries that benefit substantially from tree matching.

**+TM1<sub>JE</sub> vs Types2XExp:** I now turn my attention to the question of why +TM1<sub>JE</sub> (the best joint typed retrieval model at  $MAP=.151$  on  $Q_{FT}$ ) does not manage to outperform Types2XExp (the best textual-types only retrieval model from Section 7.1 at  $MAP=.154$  on  $Q_{FT}$ ), which does not have a formula scoring component (difference not significant).

Types2XExp (model B) produces higher ranks than +TM1<sub>JE</sub> (model A) for 61 relevant documents (from 55 queries; subset (6)). The opposite is true for 44 relevant documents (from 41 queries; subset (5)). On average, model B ranks relevant documents on  $Q_{FT}$  4,255 positions higher than model A; the difference is significant (at  $\alpha = 0.05$ ).

Figure 7.8 shows the difference in average reciprocal rank between the models across all queries in  $Q_{FT}$ . The improvements produced by model B (orange) are numerous but overall smaller than those produced by model A (blue). There are a few queries where model A manages to produce larger performance improvements than model B.

Table 7.21 shows the statistics for individual performance profile subsets, where the following differences in relevant documents are significant across subsets (5) and (6): average number of formulae (860.41 vs 624.77), number of words (5289.09 vs 4018.26) and total number of MathML nodes (22814.73 vs 14789.18).

These findings suggest that +TM1<sub>JE</sub> performs better than Types2XExp on queries where there

---

<sup>29</sup>The subset statistics table for this comparison is shown in Table D.29 of Appendix D.3.3.1. I did not find any differences in the statistics across subsets (5) and (6) to be significant.

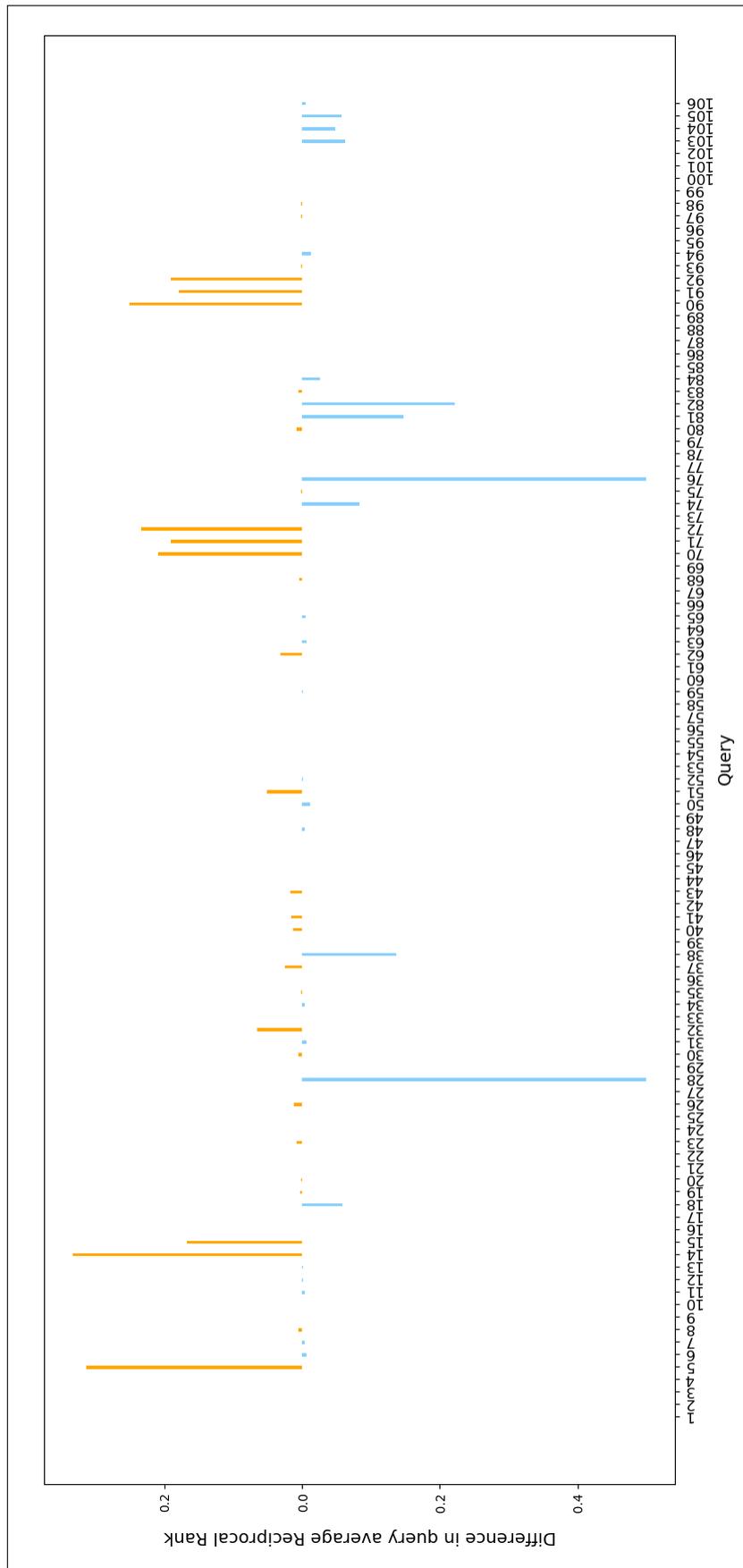


Figure 7.8: Per-query difference in reciprocal rank between +TM1<sub>JE</sub> (blue) and Types2XExp (orange) on  $Q_{FT}$ .

| Subset ID | Subset label                  | Relevant documents | Avg. No. Formulae         |              | Avg. No. Words            |              |
|-----------|-------------------------------|--------------------|---------------------------|--------------|---------------------------|--------------|
|           |                               |                    | Q                         | D            | Q                         | D            |
| (0)       | $Q_{FT}$                      | 118                | 10.63                     | 718.53       | 124.17                    | 4669.71      |
| (3)       | $A \cap B$                    | 118                | 10.63                     | 718.53       | 124.17                    | 4669.71      |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 44                 | 10.12                     | 860.41       | 120.39                    | 5289.09      |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 61                 | 11.07                     | 624.77       | 120.64                    | 4018.26      |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 13                 | 9.69                      | 678.31       | 150.15                    | 5630.15      |
| Subset ID | Subset label                  | Relevant documents | Avg. No. Types            |              | Avg. No. Positive Typings |              |
|           |                               |                    | Q                         | D            | Q                         | D            |
| (0)       | $Q_{FT}$                      | 118                | 17.83                     | 879.8        | 2.77                      | 12817505.56  |
| (3)       | $A \cap B$                    | 118                | 17.83                     | 879.8        | 2.77                      | 12817505.56  |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 44                 | 18.24                     | 973.77       | 3.07                      | 13393464.95  |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 61                 | 16.76                     | 762.79       | 2.67                      | 12858096.69  |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 13                 | 21.38                     | 1110.77      | 2.23                      | 10677638.46  |
| Subset ID | Subset label                  | Relevant documents | Avg. No. Negative typings |              | Avg. No. Typing Edges     |              |
|           |                               |                    | Q                         | D            | Q                         | D            |
| (0)       | $Q_{FT}$                      | 118                | 19.32                     | 135966583.4  | 22.09                     | 148784088.96 |
| (3)       | $A \cap B$                    | 118                | 19.32                     | 135966583.4  | 22.09                     | 148784088.96 |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 44                 | 17.85                     | 142074423.25 | 20.93                     | 155467888.2  |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 61                 | 19.55                     | 136394526.56 | 22.22                     | 149252623.25 |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 13                 | 22.62                     | 113285853.69 | 24.85                     | 123963492.15 |
| Subset ID | Subset label                  | Relevant documents | Avg. total MathML Nodes   |              | Avg. Formula Complexity   |              |
|           |                               |                    | Q                         | D            | Q                         | D            |
| (0)       | $Q_{FT}$                      | 118                | 151.02                    | 17703.15     | 14.89                     | 24.11        |
| (3)       | $A \cap B$                    | 118                | 151.02                    | 17703.15     | 14.89                     | 24.11        |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 44                 | 148.0                     | 22814.73     | 13.95                     | 25.37        |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 61                 | 155.0                     | 14789.18     | 15.58                     | 24.05        |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 13                 | 130.54                    | 14075.69     | 14.12                     | 20.1         |

Table 7.21: Subset statistics for investigating the behaviour of Types2XExp+TM1<sub>JE</sub> (model A) and Types2XExp from Section 7.1 (model B) on  $Q_{FT}$ .

is substantial formula information coming from relevant documents<sup>30</sup> (everything else being roughly equal). Model B, being agnostic of formulae, can produce better ranks than model A (subset (6)) when A does not have enough information from the formulae of relevant documents.

In this analysis we see a familiar pattern from previous analyses re-appear: tree matching models are capable of yielding substantial improvements in a few queries, but loose overall because non-tree matching models achieve many small improvements.

### 7.3.2.3 Type-aware vs Established MIR System

Lastly, I will investigate the third question: when is typed retrieval as invented here beneficial over existing formula-aware MathIR models? To this end, I will compare original Tangent (Pattaniyil and Zanibbi, 2014), VSM+UT<sup>31</sup> (model A), against my best performing typed retrieval model, +TM1<sub>JE</sub> (model B). The performance difference of MAP=.081 between the model stands in favour of +TM1<sub>JE</sub> and is significant at  $\alpha = 0.05$ <sup>32</sup> on  $Q_{FT}$ .

On average, +TM1<sub>JE</sub> ranked relevant documents 33, 610 positions higher than VSM+UT

<sup>30</sup>The higher the average number of formulae and average total of MathML nodes are, the more information can be extracted from the symbolic modality.

<sup>31</sup>Please note that this model is not identical to out-of-the box Tangent, because it benefits from my careful tuning experiments. Tangent as described by (Pattaniyil and Zanibbi, 2014) does not search the range of  $\alpha$  above 0.5 and simply sets the parameter  $\alpha$  to 0.5.

<sup>32</sup> $p=0.0108$ .

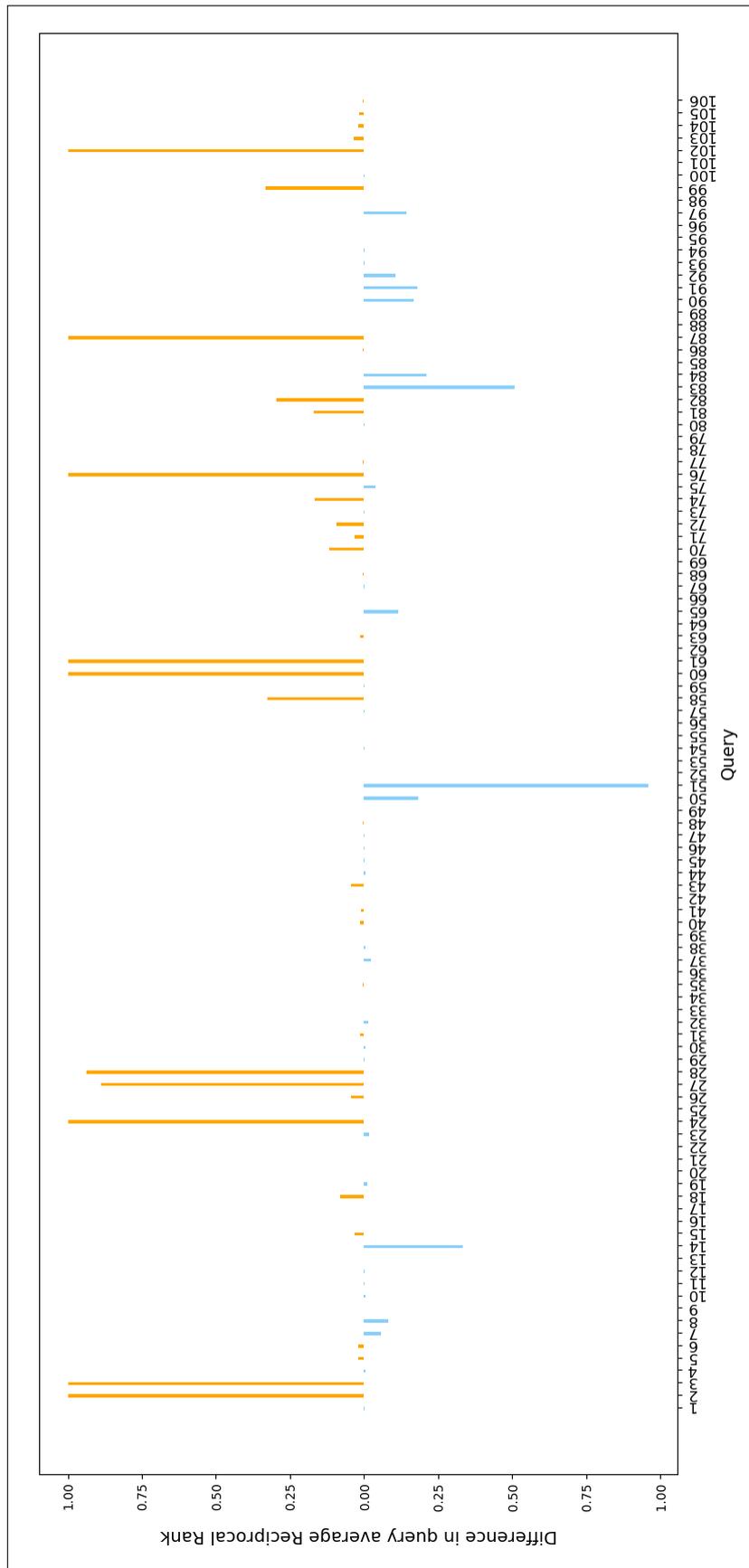


Figure 7.9: Differences in average reciprocal rank between VSM+UT (blue bars) and Types2XExp+TM1<sub>JE</sub> (orange bars) on  $Q_{FT}$ .

(difference significant). +TM1<sub>JE</sub> produced better rankings than VSM+UT for 60 relevant documents (59 queries from subset(6)) where as the opposite is true for 55 relevant documents (47 queries in subset (5)).

Figure 7.9 shows the magnitude of differences in reciprocal rank per query. We see that when +TM1<sub>JE</sub> produces better rankings than VSM+UT (orange bars), the difference is substantial (often, the difference in reciprocal is 1 or close to it). In contrast, when VSM+UT produces better rankings than +TM1<sub>JE</sub> (blue bars), VSM+UT’s advantage comes from many queries with moderately better ranks; there is one exception where the difference in reciprocal rank is close to 1 in favour of VSM+UT. The same pattern can also be observed when comparing the +TM1 and +TM1<sub>JD</sub> type-aware models to VSM+UT.

Looking at the subset statistics, shown in Table 7.22, the only statistically significant differentiator I could find between subsets (5) and (6) is the average total number of MathML nodes in relevant documents (14755.02 vs 20010.53 nodes, respectively).

| Subset ID | Subset label                  | Relevant documents | Avg. No. Formulae       |          | Avg. No. Words          |         |
|-----------|-------------------------------|--------------------|-------------------------|----------|-------------------------|---------|
|           |                               |                    | Q                       | D        | Q                       | D       |
| (0)       | $\mathbf{Q}_{FT}$             | 118                | 10.63                   | 718.53   | 124.17                  | 4669.71 |
| (3)       | $A \cap B$                    | 118                | 10.63                   | 718.53   | 124.17                  | 4669.71 |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 55                 | 11.51                   | 636.09   | 116.62                  | 4074.62 |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 60                 | 9.34                    | 774.23   | 126.22                  | 5030.42 |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 3                  | 19.67                   | 1116.0   | 185.67                  | 8365.67 |
| Subset ID | Subset label                  | Relevant documents | Avg. total MathML Nodes |          | Avg. Formula Complexity |         |
|           |                               |                    | Q                       | D        | Q                       | D       |
| (0)       | $\mathbf{Q}_{FT}$             | 118                | 151.02                  | 17703.15 | 14.89                   | 24.11   |
| (3)       | $A \cap B$                    | 118                | 151.02                  | 17703.15 | 14.89                   | 24.11   |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 55                 | 157.04                  | 14755.02 | 14.68                   | 23.54   |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 60                 | 135.25                  | 20010.53 | 14.88                   | 24.74   |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 3                  | 326.67                  | 25604.67 | 15.88                   | 21.93   |

Table 7.22: Subset statistics for VSM+UT vs VSM+TM1<sub>JE</sub> on  $\mathbf{Q}_{FT}$  (f-t-t method, 118 relevance judgements).

Overall, the number of queries in which each model performs best is roughly equal, with +TM1<sub>JE</sub> producing substantially better ranks when relevant documents have many formulae. VSM+UT is more robust when relevant documents have few formulae but most improvements in rank over +TM1<sub>JE</sub> are modest compared to the improvements observed on the subset of the corpus where the latter performs best (subset (6)).

In conclusion, +TM1<sub>JE</sub> is a substantial improvement over original Tangent (VSM+UT), while VSM+UT appears to be a robust method for formula-based retrieval, particularly when the expected amount of information coming from the symbolic modality of relevant documents is small. Although some of the improvement by +TM1<sub>JE</sub> may be attributable to the fact that it uses a stronger textual component, my analysis suggests that there are scenarios where the TM1<sub>JE</sub> typed formula retrieval component contributes more than the UT component.

### 7.3.3 Comparison to Textual Models

Finally, I compare the joint retrieval models based on Types2XExp presented in this section to the textual retrieval models (both untyped and typed) from Section 7.1. Tables 7.23, 7.24 and 7.25 show this comparison on  $Q_{FT}$ ,  $Q_F$  and  $Q$ , respectively.

|                          | Types2XExp | +TM1-           | +TM2-          | +UT             | +TT             | +TM1           | +TM1 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2           | +TM2 <sub>JD</sub> | +TM2 <sub>JE</sub> |
|--------------------------|------------|-----------------|----------------|-----------------|-----------------|----------------|--------------------|--------------------|----------------|--------------------|--------------------|
| <b>Section 7.1 model</b> | <b>MAP</b> | .147            | .069           | .133            | .130            | .140           | .150               | .151               | .142           | .096               | .106               |
| VSM                      | .072       | * <b>-.075</b>  | +0.03          | * <b>-.061</b>  | * <b>-.057</b>  | * <b>-.067</b> | * <b>-.078</b>     | ** <b>-.078</b>    | * <b>-.069</b> | -.023              | -.033              |
| BM25                     | .071       | * <b>-.076</b>  | +0.01          | * <b>-.062</b>  | -.059           | * <b>-.069</b> | * <b>-.080</b>     | * <b>-.080</b>     | * <b>-.071</b> | -.025              | -.035              |
| MLM <sub>jm</sub>        | .088       | * <b>-.059</b>  | +0.18          | * <b>-.045</b>  | * <b>-.042</b>  | * <b>-.052</b> | * <b>-.063</b>     | * <b>-.063</b>     | * <b>-.054</b> | -.008              | -.018              |
| MLM <sub>jm</sub> +RM3   | .066       | ** <b>-.081</b> | -.003          | * <b>-.067</b>  | * <b>-.064</b>  | * <b>-.074</b> | ** <b>-.085</b>    | * <b>-.085</b>     | * <b>-.076</b> | -.030              | -.040              |
| MLM <sub>dir</sub>       | .070       | ** <b>-.077</b> | +0.00          | * <b>-.063</b>  | * <b>-.060</b>  | * <b>-.070</b> | * <b>-.081</b>     | * <b>-.081</b>     | * <b>-.072</b> | -.026              | -.036              |
| MLM <sub>dir</sub> +RM3  | .081       | * <b>-.066</b>  | +0.12          | -.052           | -.048           | * <b>-.059</b> | * <b>-.069</b>     | * <b>-.069</b>     | * <b>-.061</b> | -.015              | -.024              |
| MLM <sub>dir</sub> +PRM2 | .075       | * <b>-.072</b>  | +0.06          | -.058           | -.055           | * <b>-.065</b> | * <b>-.075</b>     | * <b>-.075</b>     | * <b>-.067</b> | -.021              | -.031              |
| MLM' <sub>dir</sub>      | .070       | ** <b>-.077</b> | +0.00          | ** <b>-.063</b> | ** <b>-.060</b> | * <b>-.070</b> | * <b>-.081</b>     | * <b>-.081</b>     | * <b>-.072</b> | -.026              | -.036              |
| MLM' <sub>dir</sub> +RM3 | .036       | ** <b>-.111</b> | -.033          | ** <b>-.097</b> | * <b>-.094</b>  | * <b>-.104</b> | * <b>-.114</b>     | * <b>-.114</b>     | * <b>-.106</b> | * <b>-.060</b>     | * <b>-.070</b>     |
| SPUD                     | .095       | * <b>-.052</b>  | +0.25          | -.038           | -.035           | -.045          | * <b>-.056</b>     | -.047              | -.047          | -.001              | -.011              |
| SPUD+RM3                 | .035       | ** <b>-.112</b> | -.034          | ** <b>-.098</b> | * <b>-.094</b>  | * <b>-.105</b> | * <b>-.115</b>     | * <b>-.115</b>     | * <b>-.107</b> | * <b>-.061</b>     | * <b>-.071</b>     |
| SPUD+PRM2                | .070       | ** <b>-.077</b> | +0.01          | * <b>-.063</b>  | * <b>-.059</b>  | * <b>-.070</b> | * <b>-.080</b>     | * <b>-.080</b>     | * <b>-.072</b> | -.026              | -.036              |
| TF-IDFExp                | .047       | ** <b>-.100</b> | -.022          | ** <b>-.086</b> | * <b>-.083</b>  | * <b>-.093</b> | * <b>-.103</b>     | * <b>-.103</b>     | * <b>-.095</b> | * <b>-.049</b>     | * <b>-.059</b>     |
| Types2X                  | .073       | * <b>-.074</b>  | +0.04          | -.060           | -.057           | * <b>-.067</b> | * <b>-.078</b>     | * <b>-.078</b>     | * <b>-.069</b> | -.023              | -.033              |
| Types2XExp               | .154       | +0.07           | * <b>+.085</b> | +0.21           | +0.25           | +0.14          | +0.04              | +0.04              | +0.12          | * <b>+.058</b>     | * <b>+.049</b>     |

Table 7.23: Comparison of typed retrieval models with textual retrieval models from Section 7.1 on  $Q_{FT}$ .

|                          | Types2XExp | +TM1-           | +TM2-          | +UT             | +TT            | +TM1           | +TM1 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2           | +TM2 <sub>JD</sub> | +TM2 <sub>JE</sub> |
|--------------------------|------------|-----------------|----------------|-----------------|----------------|----------------|--------------------|--------------------|----------------|--------------------|--------------------|
| <b>Section 7.1 model</b> | <b>MAP</b> | .139            | .065           | .129            | .124           | .135           | .144               | .144               | .133           | .090               | .099               |
| VSM                      | .069       | * <b>-.070</b>  | +0.04          | * <b>-.060</b>  | * <b>-.055</b> | * <b>-.066</b> | ** <b>-.075</b>    | ** <b>-.075</b>    | * <b>-.064</b> | -.021              | -.030              |
| BM25                     | .072       | * <b>-.067</b>  | +0.07          | * <b>-.057</b>  | -.052          | * <b>-.062</b> | * <b>-.072</b>     | * <b>-.072</b>     | * <b>-.060</b> | -.018              | -.027              |
| MLM <sub>jm</sub>        | .091       | -.048           | +0.26          | -.038           | -.033          | * <b>-.044</b> | * <b>-.053</b>     | * <b>-.053</b>     | -.042          | +0.01              | -.008              |
| MLM <sub>jm</sub> +RM3   | .074       | * <b>-.065</b>  | +0.08          | * <b>-.056</b>  | -.050          | * <b>-.061</b> | * <b>-.071</b>     | * <b>-.071</b>     | * <b>-.059</b> | -.016              | -.025              |
| MLM <sub>dir</sub>       | .076       | ** <b>-.064</b> | +0.10          | * <b>-.054</b>  | -.049          | * <b>-.059</b> | ** <b>-.069</b>    | ** <b>-.069</b>    | * <b>-.057</b> | -.014              | -.023              |
| MLM <sub>dir</sub> +RM3  | .090       | -.049           | +0.25          | -.039           | -.034          | -.044          | -.054              | * <b>-.054</b>     | -.042          | +0.00              | -.009              |
| MLM <sub>dir</sub> +PRM2 | .078       | * <b>-.061</b>  | +0.13          | -.051           | -.046          | * <b>-.057</b> | * <b>-.066</b>     | * <b>-.066</b>     | -.055          | -.012              | -.021              |
| MLM' <sub>dir</sub>      | .075       | ** <b>-.065</b> | +0.09          | * <b>-.055</b>  | * <b>-.050</b> | * <b>-.060</b> | * <b>-.070</b>     | * <b>-.070</b>     | * <b>-.058</b> | -.015              | -.024              |
| MLM' <sub>dir</sub> +RM3 | .046       | ** <b>-.093</b> | -.019          | ** <b>-.083</b> | * <b>-.078</b> | * <b>-.089</b> | * <b>-.098</b>     | * <b>-.098</b>     | * <b>-.087</b> | -.044              | * <b>-.053</b>     |
| SPUD                     | .098       | -.041           | +0.33          | -.031           | -.026          | -.036          | -.046              | -.046              | -.034          | +0.08              | -.000              |
| SPUD+RM3                 | .044       | ** <b>-.095</b> | -.021          | ** <b>-.085</b> | * <b>-.080</b> | * <b>-.091</b> | * <b>-.100</b>     | * <b>-.100</b>     | * <b>-.089</b> | * <b>-.046</b>     | * <b>-.055</b>     |
| SPUD+PRM2                | .073       | ** <b>-.066</b> | +0.08          | * <b>-.057</b>  | -.051          | * <b>-.062</b> | * <b>-.071</b>     | * <b>-.072</b>     | * <b>-.060</b> | -.017              | -.026              |
| TF-IDFExp                | .053       | ** <b>-.086</b> | -.012          | ** <b>-.076</b> | * <b>-.071</b> | * <b>-.081</b> | * <b>-.091</b>     | * <b>-.091</b>     | * <b>-.079</b> | -.036              | -.045              |
| Types2X                  | .071       | * <b>-.068</b>  | +0.06          | -.058           | -.053          | * <b>-.064</b> | * <b>-.073</b>     | * <b>-.073</b>     | * <b>-.062</b> | -.019              | -.028              |
| Types2XExp               | .156       | +0.17           | * <b>+.090</b> | +0.26           | * <b>+.032</b> | +0.21          | +0.12              | +0.11              | +0.23          | * <b>+.066</b>     | * <b>+.057</b>     |

Table 7.24: Comparison of typed retrieval models with textual retrieval models from Section 7.1 on  $Q_F$ .

|                          | Types2XExp | +TM1-   | +TM2-   | +UT     | +TT     | +TM1    | +TM1 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2    | +TM2 <sub>JD</sub> | +TM2 <sub>JE</sub> |
|--------------------------|------------|---------|---------|---------|---------|---------|--------------------|--------------------|---------|--------------------|--------------------|
| <b>Section 7.1 model</b> | <b>MAP</b> | .168    | .114    | .161    | .157    | .164    | .171               | .171               | .163    | .132               | .138               |
| VSM                      | .076       | **-.092 | -.038   | **-.085 | **-.081 | **-.088 | **-.095            | **-.096            | **-.087 | *-.056             | **-.062            |
| BM25                     | .077       | **-.091 | -.037   | **-.083 | **-.080 | **-.087 | **-.094            | **-.094            | **-.086 | *-.055             | **-.061            |
| MLM <sub>jm</sub>        | .084       | **-.083 | -.030   | **-.076 | **-.073 | **-.080 | **-.087            | **-.087            | **-.079 | *-.048             | **-.054            |
| MLM <sub>jm</sub> +RM3   | .063       | **-.105 | *-.051  | **-.098 | **-.094 | **-.102 | **-.109            | **-.109            | **-.100 | **-.069            | **-.076            |
| MLM <sub>dir</sub>       | .066       | **-.102 | *-.048  | **-.095 | **-.091 | **-.099 | **-.106            | **-.106            | **-.097 | **-.066            | **-.073            |
| MLM <sub>dir</sub> +RM3  | .082       | **-.086 | -.032   | **-.079 | **-.075 | **-.083 | **-.089            | **-.090            | **-.081 | -.050              | -.057              |
| MLM <sub>dir</sub> +PRM2 | .061       | **-.106 | *-.053  | **-.099 | **-.095 | **-.103 | **-.110            | **-.110            | **-.102 | **-.071            | **-.077            |
| MLM' <sub>dir</sub>      | .072       | **-.095 | *-.042  | **-.088 | **-.085 | **-.092 | **-.099            | **-.099            | **-.091 | **-.060            | **-.066            |
| MLM' <sub>dir</sub> +RM3 | .051       | **-.117 | **-.063 | **-.110 | **-.106 | **-.114 | **-.121            | **-.121            | **-.112 | **-.081            | **-.088            |
| SPUD                     | .090       | **-.077 | -.023   | **-.070 | **-.066 | **-.074 | **-.081            | **-.081            | **-.072 | -.041              | *-.048             |
| SPUD+RM3                 | .050       | **-.118 | **-.064 | **-.111 | **-.107 | **-.114 | **-.121            | **-.121            | **-.113 | **-.082            | **-.088            |
| SPUD+PRM2                | .072       | **-.096 | *-.042  | **-.089 | **-.085 | **-.093 | **-.100            | **-.100            | **-.091 | **-.060            | **-.067            |
| TF-IDFExp                | .060       | **-.107 | *-.054  | **-.100 | **-.096 | **-.104 | **-.111            | **-.111            | **-.103 | **-.072            | **-.078            |
| Types2X                  | .070       | **-.098 | -.044   | **-.091 | **-.087 | **-.095 | **-.102            | **-.102            | **-.093 | *-.062             | **-.069            |
| Types2XExp               | .173       | +0.005  | **+.059 | +0.012  | +0.016  | +0.008  | +0.002             | +0.001             | +0.010  | **+.041            | *+.034             |

Table 7.25: Comparison of typed retrieval models with textual retrieval models from Section 7.1 on **Q**.

From Tables 7.23, 7.24 and 7.25, we observe that the Types2XExp model from Section 7.1 is numerically the best overall across all subsets of **Q**, regardless of tuning method, but statistical significance is only observed against models based on TM2. These differences are more pronounced when the f-t-t method is used for tuning<sup>33</sup>.

## 7.4 Chapter Summary

Figure 7.10 summarises the results for the best models on **Q**<sup>34</sup>, irrespective of tuning method. Bars filled with a crossed pattern indicate that the model was found to perform best when tuned on the DevSet, whereas bars filled with diagonal lines indicate that a model was found to perform best when tuned using the f-t-t method. Models that do not require tuning appear in solid bars. The thin bars around the MAP scores represent the 95% confidence intervals<sup>35</sup>(CIs) of model performances. The presented CIs indicate the ranges in which model MAPs would fall in 95% of the time if we were to repeat my experiments with data similar to that in my test collection<sup>36</sup> (Soboroff, 2014).

The first set of models, appearing in green, yellow and blue were introduced in Section 7.1. Green bars represent traditional bag-of-words retrieval models that do not treat formulae or types in any special way. The yellow bar is TFIDF-Exp from Section 7.1 which expands queries in a word-based, rather than a type-based, embedding space (as Types2XExp does)<sup>37</sup>.

<sup>33</sup>the equivalent tables with DevSet tuning can be found in Appendix D.3.4.

<sup>34</sup>The plots for **Q<sub>F</sub>** and **Q<sub>F<sub>T</sub></sub>** look very similar and are shown in Appendix D.3.5.

<sup>35</sup>Computed using the Normal, rather than the *t* distribution because the sample size is larger than 35.

<sup>36</sup>Similar in terms of queries, relevant documents, query difficulty, distribution of formulae etc.

<sup>37</sup>The reader may recall that TFDF-Exp is intended as a baseline to ascertain that any improvements observed by using the type embedding space are due to types and not due to the general approach of expanding queries using an embedding space.

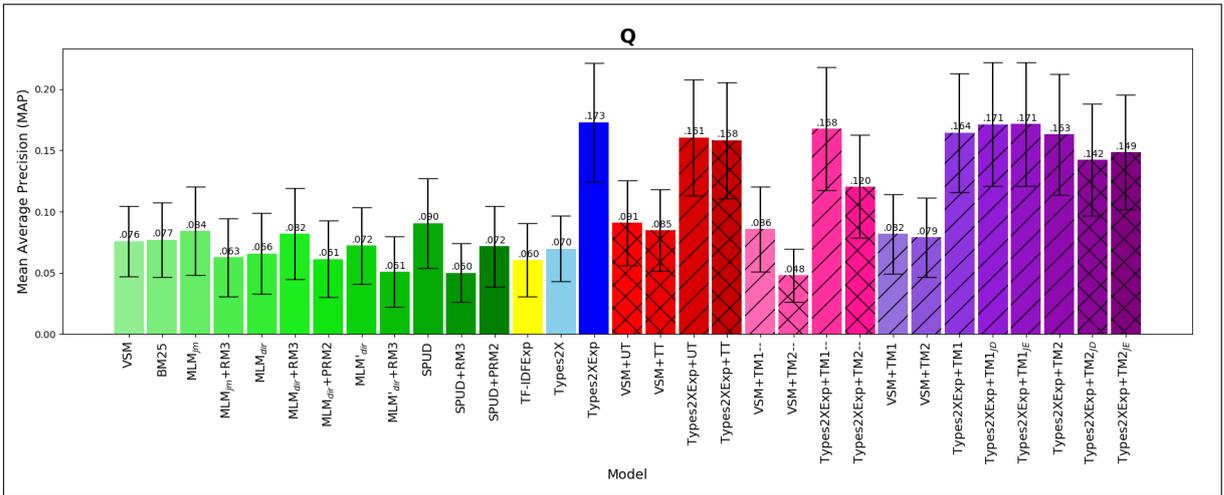


Figure 7.10: Summary of models’ retrieval results on **Q** in MAP.

The blue bars are those models that up-weight types in the text. Specifically, light blue is Types2X and blue is Types2XExp, the numerically best system in this thesis. Types2XExp extends Types2X by also performing query expansion based on types, using a type embedding space.

The remaining bars represent typed and untyped formula-based retrieval models as introduced in Sections 7.2 and 7.3: red systems represent models based on Tangent (VSM+UT, +UT and +TT, in sequence), pink systems are joint retrieval models with no semantic component (+TM1– and +TM2–) and medium purple to deep purple bars represent the full typed joint retrieval models. The performance of the best medium to deep purple systems approaches, but does not quite reach, the performance of Types2XExp from Section 7.1 (blue).

The biggest story told by Figure 7.10 is that the results support hypothesis 1, but not hypothesis 2: textual typed retrieval is effective, but joint retrieval does not add significant performance increases beyond what textual retrieval does alone.

**Hypothesis 1 confirmed:** My results show that text-based retrieval can be significantly improved by introducing types, but simply up-weighting textual types is not enough. It is the embedding space utilised in Types2XExp that really improves performance, as evidenced by the vastly superior performance of Types2XExp over Types2X (0.70 MAP on **Q**, .071 MAP on **Q<sub>F</sub>** and .073 MAP on **Q<sub>FT</sub>**).

In my experiments for Sections 7.1 and 7.2, Types2XExp from Section 7.1 emerged as the best performing model overall. On **Q** and **Q<sub>F</sub>** Types2XExp yielded .173 and .156 MAP, respectively, and performed significantly better than all models from Sections 7.1 and 7.2, including original Tangent (Pattaniyil and Zanibbi, 2014), on these subsets (cf. Tables 7.3, 7.14 and 7.15).

These results support my first hypothesis: that treating types as atomic units of information can be beneficial to MIR in the context of research-level mathematics.

**Hypothesis 2 not confirmed:** My second hypothesis is as follows:

Do retrieval models that link text and formulae using types (extracted from the text as denotations to symbols in formulae) have better retrieval efficiency over those that do not, i.e., those that treat them independently?

This hypothesis is embodied by my type unification (TM1, TM2), Tangent-based formula retrieval over types (TT) and semantic overlap on typed SLTs (TM1<sub>JD</sub>, TM1<sub>JE</sub>, TM2<sub>JD</sub>, TM2<sub>JE</sub>) models.

On  $Q_{FT}$ , where typed joint (text and formula) retrieval is best evaluated, my best typed unification models (+TM1, MAP=.140 and +TM2, MAP=.142) did not produce a consistent and statistically significant improvement in retrieval efficiency over the best untyped alternatives (UT, MAP=.133 and +TM1-, MAP=.147). In fact, as shown in Figure 7.10 and discussed earlier in this chapter, adding a type unification component to +TM1- (untyped tree matching) resulted in a small regression of retrieval performance (difference not significant).

I therefore conclude that typed unification is not as effective as I had hoped, at least as evaluated on my corpus and with my experimental setup.

However, I remain optimistic about joint typed retrieval overall for two reasons. First, +TM1<sub>JE</sub>, which is typed, was able to produce a small but very significant improvement over +TM1-, which is untyped. This may be encouraging, but in the context of this thesis +TM1<sub>JE</sub> is only a heuristic model that does not use type unification. It also does not manage to beat Types2XExp. Second, my typed tree matching models, Types2XExp+TM1<sub>JE</sub> and Types2XExp+TM1, demonstrated that they are capable of ranking relevant documents very high in a few specific cases (Section 7.3.2), and this may be an attractive feature for future models, although this positive effect is stronger for heuristic semantic overlap (+TM1<sub>JE</sub>) than it is for typed unification (+TM1).

In the future, it may well be possible to develop new typed unification and typed tree matching models that significantly reduce and offset the small (but numerous) errors in ranking exhibited by my tree matching models with additional, large improvements. After all, there are many different ways how my tree matching features could be combined. There are also many alternative scoring architectures that can be explored and this thesis has laid the ground for future research on this topic.

# Chapter 8

## Conclusions

When I started this thesis MIR was a relatively new sub-discipline of IR that was only slowly gathering steam. At that time, nobody had yet looked at research-level math retrieval by real mathematicians. Despite some early work on formula retrieval prior to 2009 (such as Miller and Youssef (2003), Kohlhase and Sucan (2006), Youssef (2007) and Kamali and Tompa (2009)), it was not until 2013 that the first shared test collections for Math IR (Aizawa et al., 2013) appeared and the IR community as a whole started paying attention to it. The initial MathIR approaches treated mathematical text like any other kind of text, with the exception of the formulae contained in it. Most of the intellectual effort was devoted to questions such as how to represent mathematical formulae in MathML (Yokoi and Aizawa, 2009; Kohlhase and Prodescu, 2013; Hambasan et al., 2014), how to match formulae using sophisticated methods such as substitution trees (Schellenberg et al., 2012) or structural heuristics (Pattaniyil and Zanibbi, 2014). At the same time, some researchers interested in both mathematics and computational linguistics started asking the kinds of questions about the connection between formulae and linguistic elements of the text that have influenced this thesis greatly (Kristianto et al., 2012; Kristianto et al., 2014b; Topic et al., 2013; Kristianto et al., 2014a; Kristianto et al., 2016; Yokoi and Aizawa, 2009).

### 8.1 Contributions

Looking back now that my thesis is finished, I have learned many things and made several contributions to science:

- I showed that it is possible to derive a test collection by observing the online dialogue between real mathematicians doing real mathematics, often achieving publications in the process. The development of a novel method for test collection creation (chapter 3), along with the careful sifting of the 160 queries that now constitute the CUMTC (Cambridge University Mathematics Test Collection) and the 13 queries in the development set required an entire year of work on my part. As a result the community now has access to this test

collection for future research, and my method of constructing a test collection can be used to create many more test collections for similar fields, machine learning tasks and search situations.

- I was influenced by scholars in theoretical linguistics who dissected the semantics of mathematical discourse (Ganesalingam, 2008), and I realised that some of their insights can be operationalised in MIR systems. The most important of these was the potential importance of mathematical types for practical math search (chapter 4), and I present an operational definition of finding such types in text.
- I developed a class of tree matching algorithms that are more flexible than pure syntactic matching and incorporate semantic matching via typed unification (chapter 6).
- Overall, I found that a special treatment of the linguistic realisation of mathematical types is beneficial to textual MIR. However, from my experiments in type-based textual retrieval (section 7.1) I learned that knowing which types are present in the queries is not enough; it is the expansion of the queries with related types that is most beneficial.
- One of my hypotheses was that the joint indexing of linguistic and symbolic realisations of types would be beneficial for MIR. This involved extensive experimentation with weighting, different linguistic indexing methods, query expansion, formula indexing and machine learning – several thousand lines of code to create indices, data pre-processing, scoring and evaluation code. As the outcome of these experiments (cf. chapter 7), we now have a better understanding of the intricacies of the task. We have also identified good candidate systems for performing MIR for research-level mathematics. One of these systems is my own contribution and is the best performing model: the Types2XExp model that expands queries with related types from the type embedding space (cf. section 7.1). Another is the heuristic system Tangent (invented by (Pattaniyil and Zanibbi, 2014)), which achieves its most competitive performance observed here only with the very specific tuning I performed here (in particular the  $\alpha$  weighting and the replacement of the default IR model, VSM, with my own Types2XExp type-based retrieval textual component). Finally, there is also the Types2XExp+TM1<sub>JE</sub> model I proposed that uses the relaxed typed tree matching method that I introduced in section 6.4.1.
- To the best of my knowledge, I am the first to ask and experimentally test hypothesis 2. Despite the fact that I have not obtained definitive experimental evidence to confirm hypothesis 2, my results inform the MIR community about the limitations and potential of typed retrieval.
- In order to realise my vision of type-based MIR I also had to define the task of automatic variable typing (chapter 5). This led me to develop and make publicly available a machine

learning data set for this task. While the dataset should hopefully be of immediate interest to researchers in MIR, it could also be useful to research outside of MIR, eg. research in mathematical topic modelling.

## 8.2 Recent Developments in MIR Test Collections

Since the development of my methodology for building test collections using MathOverflow in 2015 the MIR community has explored the community question answer (CQA) task for mathematics. The ARQMath labs at the Conference and Labs of the Evaluation Forum (CLEF) (Zanibbi et al., 2020; Mansouri et al., 2021b; Mansouri et al., 2022) introduced MIR test collections based on Math Stack Exchange (MSE) for two tasks: question retrieval and formula search.

The test collection for the math CQA task produced by the first ARQMath lab (Zanibbi et al., 2020) contained 77 questions (topics) and answers (documents) posted on MSE between 2010 and 2018 (Zanibbi et al., 2020). Relevant answers were identified using the pooling method (Section 2.4.2): five participating systems were used to rank 1,000 or fewer answer posts resulting in 500 pooled answers for each topic. Trained human assessors provided relevance assessments on a four-level scale (“Not relevant”, “Low”, “Medium” and “High” relevance). The second and third ARQMath labs (Mansouri et al., 2021b; Mansouri et al., 2022) enlarged the test collection for the two tasks. The third ARQMath lab also introduced the open domain QA task whereby questions from the CQA task can be answered by documents outside the ARQMath collection.

There are a few notable similarities in the work done by the ARQMath labs to my own test collection work with MathOverflow (Chapter 3). First, both the CUMTC and ARQMath test collections include topics that were originally posted as questions to online communities of mathematicians. Second, the four-level scale adopted by the ARQMath to assess relevance is, to some extent, analogous to my assessments for the CUMTC. For example, the definition of the relevance level “High” in Zanibbi et al. (2020) is

*Sufficient to answer the complete question on its own*

and seems to map to my own assessment of “direct” and “total” on the directness and totality dimensions, respectively (page 68). Similarly, the “Medium” level seems to be related to my own “partial” and “indirect” assessments. In my test collection construction process, I make assessments similar to the “Not relevant” and “Low” assessments made by the ARQMath labs based on the role of the citations (e.g., “Background”, “Unclear”, “Irrelevant” discussed on page 71). Third, I find the ARQMath lab organiser’s observation that some questions are better answered by textual retrieval methods while other topics are better suited for formula-based retrieval particularly of note since I have made similar observations in my own work and results. For example, in Chapter 7 I segmented my test collection in three subsets: queries with text, formulae and types ( $\mathbf{Q}$ ), queries with formulae ( $\mathbf{Q}_F$ ) and queries with at least one typed variable

( $Q_{FT}$ ) and evaluated each class of retrieval models on the most appropriate subset (as explained in Sections 7.2.1 and 7.3.1).

There are also differences between my work and the work carried-out by Zanibbi et al. (2020), Mansouri et al. (2021b) and Mansouri et al. (2022) for the ARQMath labs. One significant difference is that the CUMTC and ARQMath CQA tasks address distinct retrieval scenarios. The CUMTC is designed to evaluate retrieval of research-level mathematics with research papers in mathematics as the retrieval unit where as the ARQMath CQA task is designed for retrieval of specific answers to mathematical questions. Furthermore, questions posted on Math Stack Exchange represent mathematical problems of lower difficulty than those posted on MathOverflow (as discussed in Section 3.2.2).

Differences between my test collection construction process and the process adopted by the ARQMath labs can also be identified. One difference is that my process does not use pooling: I collected candidate relevant documents by observing expert recommendations and delegated relevance decisions to the questioner as the ultimate judge of relevance (page 75). Another difference is that my approach requires considerably less effort on the part of the test collection builder because there is no large set of pooled candidate relevant documents to be manually assessed.

Despite the differences between the CUMTC and ARQMath CQA test collections and underlying retrieval scenarios, I am excited to see that some of own my test collection decisions (e.g., similar relevance assessment scales and desirable thread characteristics) and observations were also discovered by other researchers for similar tasks.

### 8.3 Limitations and future work

After finishing a PhD thesis, people typically have ideas of what they would have done differently and how they would want to take the work further. As to the first of these, with hindsight I wished I had built a bigger development set for my test collection. As the experiments showed, tuning the values of the linear combination parameter  $\alpha$  on a bigger development set would very likely have given me more stable results. I therefore ended up using the f-t-t method as an alternative. A larger corpus would have also served to see if the spikes observed by tree matching models are part of a recurring phenomenon in MIR (or might even be particular to research-level MIR).

It is disappointing that the sophisticated tree models (and in particular the operation of typed unification) that I implemented in section 6.2 were not as successful as I had hoped. This sparks three possible extensions of the work presented in this thesis:

- Graph neural networks have become popular during the time I worked on the thesis. There are reasons to believe that they could ideally represent formulae and thus enrich my work.
- It is a natural question to ask whether type unification can be useful in MIR if used with other formula matching paradigms, such as Tangent heuristic formula retrieval.

- Another recent development is the emergence of learn-to-rank models. I will argue why they might be perfectly suited to achieve better MIR performance in my situation.

### 8.3.1 Graph Neural Network Representations

Recently formula representation using embeddings has become a topic of interest in the MIR community. For example, Mansouri et al. (2019) experimented with representing formulae using vector embeddings constructed from SLTs and Pfahler and Morik (2020) explored methods for training formula embeddings using Graph Convolutional Networks (GCNs), a special instance of Graph Neural Networks (GNNs) (Scarselli et al., 2009; Micheli, 2009).

There are various ways GNNs could be applied to my task of research-level mathematics retrieval. A Typed SLT can be viewed as a graph, with typed terminals and typed non-terminals as the nodes and positional links (e.g., ABOVE and BELOW) as the directed edges between them.

Each node is assigned a feature matrix that may be composed of rich representations for (a) the symbols occurring in the node, (b) the types assigned to the node through variable typing, (c) a representation for the kind of node (e.g., `LayoutFraction`). Features for types and symbols can take the form of term or embedding vectors while features for the kind of node can be incidence vectors over the dictionary of terminal and non-terminal labels or label embeddings constructed from the SLTs in the MREC. There are many node and edge features I could use with many suitable representations and it is straightforward to produce from data I already have.

Representation learning could then be applied to learn the best graph embedding for each SLT. Scoring would take the form of computing the cosine similarity between SLT embedding vectors. I believe this approach could be beneficial to MIR because GNNs can learn to jointly represent the topological structure of formulae and the distribution of features (types) of their terminals and non-terminals (Hamilton et al., 2018). Retrieval models utilising learned formula representations can be developed with DeepWalk (Perozzi et al., 2014), Node2Vec (Grover and Leskovec, 2016) and GraphSAGE (Hamilton et al., 2018) as their basis and evaluated on the CUMTC.

Of course, to do this I will have to build a fresh data set for training formula representations tailored to research-level mathematics. However, building this data set will not be as costly as building my test collection from scratch: I now have a process I can apply (Chapter 3).

### 8.3.2 Typed Tangent with Unification

Experience from my previous experiments suggests that Tangent’s heuristic of breaking-up the structure of formulae into a set of tuples exhibits performance characteristics that I can exploit.

Tangent describes the structure of formulae using a set of tuples that record symbols and their positional relationship. I also know from my experiments that retrieval with types can be beneficial, although I have not achieved significance in the case of typed formulae. It is therefore natural to ask whether there is a way to combine the retrieval advantages of types with those of

the Tangent heuristic. My first typed Tangent model (TT from section 6.4.3) replaced symbols with types when possible (i.e., when types can be found in the text), but it turns out that mixing types and symbols in a global Tangent table does not work well, as shown by TT’s relatively low performance.

My next attempt at typing Tangent could try to implement the following ideas of modeling additional levels of similarity between formulae:

- Match on unifiable types instead of identical types; use type unification to do so (fields with type labels could be said to match, even if the labels are not identical, if the types can be unified using the `unify` operation)
- Match partially in cases of partially unknown types in formulae
- Index and match on skolemised forms

This could be done by introducing parallel Tangent indexes, one for types and one for skolemised forms, which are paired with the tuples in the original Tangent index. This pairing between typed and original Tangent tuples would allow me to model formula similarity based on both syntax and types. The original Tangent scoring metrics for a pair of formulae (described in Section 2.2.5) can then be modified so that the quantification of similarity takes both original and typed tuples into account.

Let us recall the structure of Tangent tuples and how query and document tuples are matched. A Tangent tuple consists of four fields. The `parent` and `child` fields record the start and end symbols of any directed edge in an SLT, respectively. The remaining two fields encode the spatial relationship between the parent and child: the `Dist` field records the cumulative distance (total number of edges) between parent and child, while `Vert` records the vertical (baseline) distance between them. The left column of Figure 8.1 shows how the formula  $x + y$  is decomposed into 5 tuples in an original Tangent index.

| Original Tangent index |       |       |       | Typed Tangent index           |          |       |   |
|------------------------|-------|-------|-------|-------------------------------|----------|-------|---|
| $x + y$                |       |       |       | $x + y; y :: \text{“number”}$ |          |       |   |
| Parent                 | Child | Dist. | Vert. | Parent                        | Child    | Dist. |   |
| x                      | +     | 1     | 0     | *                             | *        | 1     | 0 |
| x                      | y     | 2     | 0     | *                             | “number” | 2     | 0 |
| x                      | +     | 3     | 0     | *                             | *        | 3     | 0 |
| +                      | y     | 1     | 0     | *                             | “number” | 1     | 0 |
| y                      | None  | 0     | 0     | “number”                      | None     | 0     | 0 |

Figure 8.1: Document Index: original (left column) and one form of typed (right column) Tangent indices for the formula:  $x + y; y :: \text{“number”}$ .

In original Tangent, two tuples, one from the query and another from a document, match if all of their fields are identical field-by-field: the parent symbol in the query tuple is the same as the parent symbol in the document tuple and so on.

In the typed index, presented on the right column of Figure 8.1, we see how  $x+y$ ,  $y :: \text{"number"}$  is represented; in particular, the type of  $y$  is recorded as number ( $y :: \text{"number"}$ ), whereas the type of  $x$  in this example is unknown, shown here by the unknown type marker  $**$ .

Suppose the index shown in Figure 8.1 is the document index (one document with one formula,  $x + y$ ,  $y :: \text{"number"}$ ).

Suppose also that a query with one formula  $x + y$ ;  $x, y :: \text{"number"}$ <sup>1</sup> is issued to the typed index in Figure 8.1.

The tuples for the query are shown in Figure 8.2.

| Original Tangent index |       |       |       | Typed Tangent index              |          |       |   |
|------------------------|-------|-------|-------|----------------------------------|----------|-------|---|
| $x + y$                |       |       |       | $x + y$ ; $y :: \text{"number"}$ |          |       |   |
| Parent                 | Child | Dist. | Vert. | Parent                           | Child    | Dist. |   |
| x                      | +     | 1     | 0     | "number"                         | "number" | 1     | 0 |
| x                      | y     | 2     | 0     | "number"                         | "number" | 2     | 0 |
| x                      | +     | 3     | 0     | "number"                         | *        | 3     | 0 |
| +                      | y     | 1     | 0     | *                                | "number" | 1     | 0 |
| y                      | None  | 0     | 0     | "number"                         | None     | 0     | 0 |

Figure 8.2: Query Index: original (left column) and one form of typed (right column) Tangent indices for the formula:  $x + y$ ;  $x, y :: \text{"number"}$ .

We can see from the figure that the fields of two typed tuples from the query are direct matches to two corresponding tuples in the document (rows 4 and 5 respectively).

But all entries for typed tuple pairs for "x" are not direct matches, and intuitively a partial match seems appropriate for this situation. The question then becomes to what degree the typed tuples in the query and document should match. Let us look at two tuples from the document index and the query as an example:

(\*, "number", 2, 0) (document tuple)  
 ("number", "number", 2, 0) (query tuple)

I propose that for this particular tuple pair, a partial match score of  $\frac{3}{4}$  should be assigned, as three out of four fields match (child, horizontal and vertical distance).

Another minor change along the same lines is whether the syntactic similarity of formulae could be further boosted by taking skolemisations into account; this could be investigated by adding a further paired index for skolemised formulae to the typed Tangent model. The skolemised Tangent index would store the skolems instead of raw symbols; matching of skolemised tuples can work in the same way as in original Tangent.

Implementing such ideas would put me in a position to experimentally investigate the research question concerning the compatibility of Tangent's heuristic approach with type information. In my opinion, there are good reasons to believe that these modifications would indeed enable

<sup>1</sup>Notation reminder:  $x, y :: \text{"number"}$  means that both  $x$  and  $y$  are typed as numbers.

a typed version of Tangent to perform well. I believe the modifications I proposed introduce a form of relaxed matching while preserving the ideal of not penalising syntactic similarity scores. They do so in a better way than TT did.

In my TT model I replaced symbols with types, leaving the symbols in place if no type could be found. This approach can introduce penalties to the Tangent heuristics because identical query–document formulae might seem to have mismatching tuples when in fact the overlap of typable symbols between them is small or empty. The resulting false negative tuple matches also implicitly penalise the structural similarity of the formulae because it is the types, not the symbols or distance heuristics that failed to match.

Separating type and symbol matching into parallel indices resolves this problem and is the best of both worlds: we can still apply flexible structural matching with the original Tangent index but also be specific in what the symbols mean through their types.

### 8.3.3 Learn-to-rank Models

Experiments conducted by Mansouri et al. (2021a) suggest that learning-to-rank (LTR) from multiple formula retrieval methods can be beneficial to MIR. Similar approaches might also be beneficial to the models presented in this thesis and their derivatives, for the particular task of research-level mathematics.

LTR is a class of supervised machine learning methods in IR (Li, 2011a). The objective is to learn how to rank relevant documents in response to queries from data, rather than creating hand-crafted ranking functions like BM25. In general, training and testing learning-to-rank models involves projecting query–document pairs onto a feature space and learning how to produce an ordered permutation of documents such that the deviation from the optimal permutation (the relevant documents appearing at the top ordered by grade) is minimised. What I described here is the global ranking model but a local ranking model that learns how to score individual query–document pairs is also possible (LI, 2011b). In my context, LTR could be used to build better models that aggregate the rankings produced by the various models proposed in this thesis, as was done by Mansouri et al. (2021a). The idea of learning to aggregate is not new: Cranking is a method that learns the conditional probability of the ranking of documents from multiple ranked lists, Lebanon and Lafferty (2002), whereas Borda counts (Aslam and Montague, 2001) create a new ranked list with each document sorted according to the number of documents below it in the base-system ordered lists. These systems would present plausible baselines for my new task.

Other opportunities for LTR experiments in research-level MIR use the tree matching feature set more directly. I am particularly interested in investigating the usefulness of my tree matching features using an SVM MAP (Yue et al., 2007) LTR model. In this setup, my tree matching features become input feature vectors for query–document pairs to a structural SVM classifier optimised directly on MAP.

Data for training learning-to-rank models come from a test collect with graded relevance

judgements (LI, 2011b) or clickthrough logs (Joachims and Thorsten, 2002). In order to train and evaluate learning-to-rank models for research mathematics I would need to re-apply my test collection construction method to create a train-test-evaluate machine learning data set for this task. This should not be too difficult, because I have accounted for the possibility of graded relevance judgements while developing my test collection construction method. Furthermore, I have already produced graded relevance judgements while building my test collection.

### 8.3.4 Refined Type-Sense Disambiguation and Modelling

I modelled subtype/supertype relationships between types using a suffix trie and similarity in a type embedding space. My type disambiguation and unification algorithms (Sections 6.3.1 and 6.2, respectively) implement type relatedness by coercing types to a common supertype on the known type suffix trie (KTST, Chapter 4) or implicitly by finding types that are similar to both types in the type embedding space.

Since I completed my experimental work with Types2XExp Youssef and Miller (2019) also explored the use of word embeddings for query expansion in MIR. The authors used the centroid of the embedding vectors for the input search words to discover related words from the word embedding space they constructed. Youssef and Miller (2019) found that the trained embedding space returned words that were topically related to the input words, like I did, but only experimented with atomic words and not multi-phrase types. In Section 7.1 I demonstrated that types as atomic tokens are more beneficial to MIR than their constituent words. I would, therefore, like to conduct a more detailed investigation to determine how well my type embedding space models the subtype/supertype relationship between types.

I trained my type embeddings using Word2Vec (Mikolov et al., 2011; Mikolov et al., 2013d; Mikolov et al., 2013b) to create a unique, *global vector representation* for each type. However, global type embedding vectors do not explicitly model the different context-specific senses of words (Liu et al., 2020) or types. Therefore, I would like to investigate the effectiveness of the type embedding space in identifying instances of types that are

- syntactically equivalent but are polysemous (i.e., have multiple related meanings depending on the context).
- synonymous but have no common supertype on the surface (i.e., semantically equivalent);
- syntactically equivalent (i.e., the same string) but are homonyms (i.e., have two or more unrelated senses).

in a dedicated experiment. As an extension to this experiment I can build and evaluate a new type embedding space using *contextualised word embeddings* – context-dependent representations that capture semantic and syntactic properties of words across linguistic contexts (Liu et al., 2020; Peters et al., 2018). Contextualised embeddings are generally regarded as good models for polysemy (Peters et al., 2018; Mun and Desagulier, 2022; Yenicelik et al., 2020) and have shown promising results in homonymy detection (Garcia, 2021).

In future work I can also investigate the use of machine learning classifiers to predict the kind of relationship between a source type and candidate related types (e.g., during type disambiguation and unification). In this context labels such as “supertype”, “subtype”, “topical similarity”, “synonym” and “no relation” can be used to characterise type relationships. Contextualised embeddings and large lexical databases for mathematics (Shan and Youssef, 2021) can be used to derive input features and to carry out evaluation. Type relationship classifiers can be used to make more fine-grained type disambiguation decisions and to decide whether two typed variables should be unified in a particular context.

### 8.3.5 Refined Type Disambiguation

Finally, type disambiguation is a very important step in typed retrieval and there are many possibilities on how types for variables could be disambiguated that I have left unexplored in this thesis.

I used a simple scoring method to select the best type for each variable from a list of possible typings aggregated from all sentences in the document. A next iteration in my research could be to investigate replacing the scoring method with machine learning models.

We know from Section 2.5 that symbols and notation are interspersed in the discourse and embedded in the textual modality of mathematics (Ganesalingam, 2008). We also know from research conducted by Wolska and Grigore (2010) and Wolska et al. (2011) that a large percentage of symbols are declared and used near their first introduction. Therefore, I would like to investigate if replacing the document-wide approach to type disambiguation<sup>2</sup> with a more localised one would have positive downstream effects to typed retrieval. For example, I could investigate models for performing type disambiguation of SLT instances from typings localised to a window of sentences surrounding them.

I could also experiment with new and smaller units of retrieval, such as sections in the document. Mathematicians are known to re-use variables by changing their types across sections, and experiments at that level would allow me to see if modeling variable typings per section is detrimental or beneficial overall.

In general, it is reasonable to expect that the choice of type disambiguation strategies should affect Math IR with typed retrieval. An important objective for my future work is to conduct experiments with many strategies to confirm and quantify the overall effect of type disambiguation to typed retrieval.

Looking back at the entirety of this work, I want to reiterate that retrieval of research-level mathematics is a difficult task. However, I hope that the work I have done for my research has made the task more accessible to the wider IR community and has laid the foundation for continued research in retrieval models for research mathematics.

---

<sup>2</sup>Recall that typings for each variable were first collected from the entire document before type disambiguation is applied.

## References

- Nasreen Abdul-jaleel, James Allan, W. Bruce Croft, O Diaz, Leah Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. 2004. Umass at trec 2004: Novelty and hard. In *In Proceedings of TREC-13*.
- Alfred V. Aho and Margaret J. Corasick. 1975. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340, June.
- J. Aitchison and C.W. Cleverdon. 1963. *A Report on a Test of the Index of Metallurgical Literature of Western Reserve University*. Aslib Cranfield Research Project. Aslib. Cranfield Research Project, College of Aeronautics.
- Akiko Aizawa, Michael Kohlhase, and Iadh Ounis. 2013. Ntcir-10 math pilot task overview. In *Proceedings of the 10th NTCIR Conference*, June.
- Akiko Aizawa, Michael Kohlhase, Iadh Ounis, and Moritz Schubotz. 2014. NTCIR-11 math-2 task overview. In *Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-11, National Center of Sciences, Tokyo, Japan, December 9-12, 2014*.
- Javed A. Aslam and Mark Montague. 2001. Models for metasearch. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '01*, page 276–284, New York, NY, USA. Association for Computing Machinery.
- Peter Bailey, Nick Craswell, Ian Soboroff, Paul Thomas, Arjen de Vries, and Emine Yilmaz. 2008. Relevance assessment: are judges exchangeable and does it matter. pages 667–674, 01.
- Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92*, pages 144–152, New York, NY, USA. ACM.
- Chris Buckley and Ellen M. Voorhees. 2000. Evaluating evaluation measure stability. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '00*, pages 33–40, New York, NY, USA. ACM.
- Chris Buckley and Ellen M. Voorhees. 2004. Retrieval evaluation with incomplete information. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '04*, pages 25–32, New York, NY, USA. ACM.
- Robert Burgin. 1992. Variations in relevance judgments and the evaluation of retrieval performance. *Inf. Process. Manage.*, 28(5):619–627, July.
- Sangbum Cho and Darryl McCullough. 2006. The tree of knot tunnels. *arXiv Mathematics e-prints*, page math/0611921, Nov.
- Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling.
- C.W. Cleverdon, J. Mills, and M. Keen. 1966. *Factors Determining the Performance of Indexing Systems, Vol. 1: Design*. Number v. 1 in *Factors Determining the Performance of Indexing Systems*. College of Aeronautics.

- C. W. Cleverdon. 1960. Report on the first stage of an investigation into the comparative efficiency of indexing systems. Technical report.
- C. W. Cleverdon. 1962. Aslib cranfield research project: Report on the testing and analysis of an investigation into the comparative efficiency of indexing systems. Technical report.
- Cyril Cleverdon. 1967. The cranfield tests on index language devices. *Aslib Proceedings*, 19(6):173–194.
- Cyril W. Cleverdon. 1991. The significance of the cranfield tests on index languages. In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '91, pages 3–12, New York, NY, USA. ACM.
- P. Clough and M. Sanderson. 2013. Evaluating the performance of information retrieval systems using test collections. *Information Research*, 18(2), June. © Year The Author(s). This is an Open Access article distributed under the terms of the Creative Commons Attribution Licence (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.
- W.S. Cooper. 1971. A definition of relevance for information retrieval. *Information Storage and Retrieval*, 7(1):19 – 37.
- W. S. Cooper. 1972. The inadequacy of probability of usefulness as a ranking criterion for retrieval system output. *Unpublished Working Paper*.
- Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September.
- Ronan Cummins, Jiaul H. Paik, and Yuanhua Lv. 2015. A pÓlya urn document language model for improved information retrieval. *ACM Trans. Inf. Syst.*, 33(4):21:1–21:34, May.
- Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Hubbard, L. D. Jackel, and D. Henderson, 1990. *Handwritten Digit Recognition with a Back-Propagation Network*, page 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Reinhard Diestel. 2012. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer.
- David Dubin. 2004. The most influential paper gerard salton never wrote. *Library Trends*, 52(4):748–764.
- John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159.
- Sir Fisher, Ronald Aylmer. 1970. *Statistical methods for research workers*. Edinburgh : Oliver and Boyd, 14th ed., revised and enlarged edition. Previous ed. 1958.
- Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many rater. *Psychological Bulletin*, 76:378–382.
- J.B. Fraleigh. 2003. *A First Course in Abstract Algebra*. World student series. Addison-Wesley, 7th international edition edition.

- Katerina T. Frantzi, Sophia Ananiadou, and Jun-ichi Tsujii. 1998. The c-value/nc-value method of automatic recognition for multi-word terms. In *Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries, ECDL '98*, pages 585–604, London, UK, UK. Springer-Verlag.
- Atsushi Fujii, Makoto Iwayama, and Noriko K. 2006. Test collections for patent retrieval and patent classification in the fifth ntcir workshop.
- Takahiro Fukusima and Manabu Okumura. 2001. Text summarization challenge: Text summarization evaluation at ntcir workshop2. *SIGIR Forum*, 38:29–38.
- Axel Gandy. 2009. Sequential implementation of monte carlo tests with uniformly bounded resampling risk. *Journal of the American Statistical Association*, 104(488):1504–1511.
- Mohan Ganesalingam. 2008. *The Language of Mathematics*. Ph.D. thesis, Cambridge University Computer Laboratory.
- Marcos Garcia. 2021. Exploring the representation of word meanings in context: A case study on homonymy and synonymy. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3625–3640, Online, August. Association for Computational Linguistics.
- David Gauld. 2009. Metrisability of Manifolds. *arXiv preprint arXiv:0910.0885*.
- Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr. PMLR.
- Winfried Gödert. 2012. Detecting multiword phrases in mathematical text corpora. *CoRR*, abs/1210.0852.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- E. Graf and L. Azzopardi. 2008. A methodology for building a patent test collection for prior art search.
- Peter Graf. 1994. Substitution tree indexing. Research Report MPI-I-94-251, Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany, October.
- Mihai Grigore, Magdalena Wolska, and Michael Kohlhase. 2009. Towards context-based disambiguation of mathematical expressions. In *The joint conference of ASCM 2009 and MACIS 2009. 9th international conference on Asian symposium on computer mathematics and 3rd international conference on mathematical aspects of computer and information sciences, Fukuoka, Japan, December 14–17, 2009. Selected papers.*, pages 262–271. Fukuoka: Kyushu University, Faculty of Mathematics.
- Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks.

- Ferruccio Guidi and Claudio Sacerdoti Coen. 2015. A survey on retrieval of mathematical knowledge. *CoRR*, abs/1505.06646.
- Radu Hambasan, Michael Kohlhase, and Corneliu-Claudiu Prodescu. 2014. Mathwebsearch at NTCIR-11. In *Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-11, National Center of Sciences, Tokyo, Japan, December 9-12, 2014*.
- William L. Hamilton, Rex Ying, and Jure Leskovec. 2018. Inductive representation learning on large graphs.
- Donna Harman. 1993. Overview of the first trec conference. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '93*, pages 36–47, New York, NY, USA. ACM.
- Donna Harman. 1994. Overview of the second text retrieval conference (trec-2). In *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 1–20.
- Donna Harman. 1995. Overview of the third text retrieval conference (trec-3). In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 1–20.
- Donna Harman. 2011. *Information Retrieval Evaluation*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers.
- Stephen P. Harter. 1975. A probabilistic approach to automatic keyword indexing. part i. on the distribution of specialty words in a technical literature. *Journal of the American Society for Information Science*, 26(4):197–206.
- Stephen P. Harter. 1996. Variations in relevance assessments and the measurement of retrieval effectiveness. *Journal of the American Society for Information Science*, 47(1):37–49.
- David Hawking and Nick Craswell. 2004. Very large scale retrieval and web search.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997a. Long Short-term Memory. *Neural Computation*, 9.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997b. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Daisuke Ishikawa, Tetsuya Sakai, and Noriko Kando. 2010. Overview of the ntcir-8 community qa pilot task (part i): The test collection and the task.
- Makoto Iwayama, Atsushi Fujii, Noriko Kando, and Akihiko Takano. 2003. Overview of patent retrieval task at ntcir-3. In *Proceedings of the ACL-2003 Workshop on Patent Corpus Processing - Volume 20, PATENT '03*, pages 24–32, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Bernard J. Jansen, Amanda Spink, Judy Bateman, and Tefko Saracevic. 1998. Real life information retrieval: A study of user queries on the web. *SIGIR Forum*, 32(1):5–17, April.
- Kevin Jarrett, Koray Kavukcuoglu, Marc’Aurelio Ranzato, and Yann LeCun. 2009. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153.

- Joachims and Thorsten. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 133–142, New York, NY, USA. Association for Computing Machinery.
- Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 2 edition.
- John S. Justeson and Slava M. Katz. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(1):9–27.
- Shahab Kamali and Frank Wm. Tompa. 2009. Improving mathematics retrieval. In *Towards a Digital Mathematics Library. Grand Bend, Ontario, Canada, July 8-9th, 2009*, pages 37–48. Masaryk University Press.
- Shahab Kamali and Frank Wm. Tompa. 2010. A new mathematics retrieval system. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1413–1416, New York, NY, USA. ACM.
- Shahab Kamali and Frank Wm. Tompa. 2013. Structural similarity search for mathematics retrieval. In Jacques Carette, David Aspinall, Christoph Lange, Petr Sojka, and Wolfgang Windsteiger, editors, *Intelligent Computer Mathematics*, pages 246–262, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Noriko Kando, Kazuko Kuriyama, Toshihiko Nozue, Koji Eguchi, Hiroyuki Kato, Soichiro Hidaka, and Jun Adachi. 1999. The ntcir workshop : the first evaluation workshop on japanese text retrieval and cross-lingual information retrieval.
- Tsuneaki Kato and Fumito Masui. 2003. Question answering challenge (qac-1) an evaluation of question answering task at ntcir workshop 3. 02.
- R.V. Katter. 1968. The influence of scale form on relevance judgments. *Information Storage and Retrieval*, 4(1):1 – 11.
- Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Michael Kohlhase and Corneliu-Claudiu Prodescu. 2013. Mathwebsearch at NTCIR-10. In *Proceedings of the 10th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-10, National Center of Sciences, Tokyo, Japan, June 18-21, 2013*.
- Michael Kohlhase and Ioan A. Sucan. 2006. A search engine for mathematical formulae. In *Proc. of Artificial Intelligence and Symbolic Computation, number 4120 in LNAI*, pages 241–253. Springer.
- Giovanni Yoko Kristianto, Minh quoc Nghiem, Yuichiroh Matsubayashi, and Akiko Aizawa. 2012. Extracting definitions of mathematical expressions in scientific papers. In *In JSAI*.
- Giovanni Yoko Kristianto, Florence Ho, and Akiko Aizawa. 2014a. The mcat math retrieval system for ntcir-11 math track. In *In Proc. NTCIR Workshop 11 Meeting*.
- Giovanni Yoko Kristianto, Goran Topic, and Akiko Aizawa. 2014b. Exploiting textual descriptions and dependency graph for searching mathematical expressions in scientific papers.

- Giovanni Yoko Kristianto, Goran Topic, and Akiko Aizawa. 2016. MCAT math retrieval system for NTCIR-12 mathir task. In *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies, National Center of Sciences, Tokyo, Japan, June 7-10, 2016*.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of NAACL-HLT 2016*.
- R. Ray Larson, Reynolds J. Choe, and Fredric C Gey. 2013. The abject failure of keyword ir for mathematics search: Berkeley at ntcir-10 math. In *In Proceedings of the 10th NTCIR conference*.
- Guy Lebanon and John D. Lafferty. 2002. Cranking: Combining rankings using conditional probability models on permutations. In *ICML*.
- W.C. Lee and E.A. Fox. 1988. *Experimental Comparison of Schemes for Interpreting Boolean Queries*. TR (Virginia Polytechnic Institute and State University. Department of Computer Science). Department of Computer Science, Virginia Polytechnic Institute and State University.
- Hang Li. 2011a. Learning to rank for information retrieval and natural language processing. In *Synthesis Lectures on Human Language Technologies*.
- Hang LI. 2011b. A short introduction to learning to rank. *IEICE Transactions on Information and Systems*, E94.D(10):1854–1862.
- Martin Lísška, Petr Sojka, and Michal Růžička. 2013. Similarity search for mathematics: Masaryk university team at the ntcir-10 math task. In Kazuaki Kishida Noriko Kando, editor, *Proceedings of the 10th NTCIR Conference on Evaluation of Information Access Technologies*, pages 686–691, Tokyo. National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430 Japan.
- Qi Liu, Matt J. Kusner, and Phil Blunsom. 2020. A survey on contextual embeddings. *CoRR*, abs/2003.07278.
- Martin Lísška, Petr Sojka, Michal Růžička, and Petr Mravec. 2011. Web interface and collection for mathematical retrieval: Webmias and mrec. In Petr Sojka and Thierry Bouche, editors, *Towards a Digital Mathematics Library.*, pages 77–84, Bertinoro, Italy, Jul. Masaryk University.
- Antonio Lorenzin. 2020. Uniqueness of triangulated categories with a heart of dimension 0 or 1.
- Yuanhua Lv and ChengXiang Zhai. 2009. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09*, pages 1895–1898, New York, NY, USA. ACM.
- Yuanhua Lv and ChengXiang Zhai. 2011. Lower-bounding term frequency normalization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11*, pages 7–16, New York, NY, USA. ACM.
- Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

- Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.
- Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.
- Behrooz Mansouri, Shaurya Rohatgi, Douglas Oard, Jian Wu, C. Lee Giles, and Richard Zanibbi. 2019. Tangent-cft: An embedding model for mathematical formulas. 07.
- Behrooz Mansouri, Richard Zanibbi, and Douglas W. Oard, 2021a. *Learning to Rank for Mathematical Formula Retrieval*, page 952–961. Association for Computing Machinery, New York, NY, USA.
- Behrooz Mansouri, Richard Zanibbi, Douglas W. Oard, and Anurag Agarwal. 2021b. Overview of arqmath-2 (2021): Second clef lab on answer retrieval for questions on math. In K. Selçuk Candan, Bogdan Ionescu, Lorraine Goeriot, Birger Larsen, Henning Müller, Alexis Joly, Maria Maistro, Florina Piroi, Guglielmo Faggioli, and Nicola Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 215–238, Cham. Springer International Publishing.
- Behrooz Mansouri, Anurag Agarwal, Douglas W. Oard, and Richard Zanibbi. 2022. Advancing math-aware search: The arqmath-3 lab at clef 2022. In Matthias Hagen, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørkvåg, and Vinay Setty, editors, *Advances in Information Retrieval*, pages 408–415, Cham. Springer International Publishing.
- M. E. Maron and J. L. Kuhns. 1960. On relevance, probabilistic indexing and information retrieval. *J. ACM*, 7(3):216–244, July.
- Ursula Martin and Alison Pease. 2013. What does mathoverflow tell us about the production of mathematics? *CoRR*, abs/1305.0904.
- Alessio Micheli. 2009. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511.
- Tomas Mikolov, Stefan Kombrink, Lukáš Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*, pages 5528–5531.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.
- Tomáš Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013b. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013c. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

- Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013d. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June. Association for Computational Linguistics.
- Bruce R. Miller and Abdou Youssef. 2003. Technical aspects of the digital library of mathematical functions. *Annals of Mathematics and Artificial Intelligence*, 38(1):121–136, May.
- S. Mizzaro. 1997. Relevance: The whole story. *Journal of the American Society for Information Science*, (48):810–832.
- Seongmin Mun and Guillaume Desagulier. 2022. How do transformer-architecture models address polysemy of Korean adverbial postpositions? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 11–21, Dublin, Ireland and Online, May. Association for Computational Linguistics.
- Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10*, page 807–814, Madison, WI, USA. Omnipress.
- Minh-Quoc Nghiem, Giovanni Yoko Kristianto, Goran Topić, and Akiko Aizawa. 2014. Which one is better: Presentation-based or content-based math search? In Stephen M. Watt, James H. Davenport, Alan P. Sexton, Petr Sojka, and Josef Urban, editors, *Intelligent Computer Mathematics*, pages 200–212, Cham. Springer International Publishing.
- Joakim Nivre. 2010. Dependency parsing. *Language and Linguistics Compass*, 4(3):138–152.
- Nidhin Pattaniyil and Richard Zanibbi. 2014. Combining TF-IDF text retrieval with an inverted index over symbol pairs in math expressions: The tangent math search engine at NTCIR 2014. In *Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-11, National Center of Sciences, Tokyo, Japan, December 9-12, 2014*.
- Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, pages 701–710, New York, NY, USA. ACM.
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.
- Lukas Pfahler and Katharina Morik, 2020. *Semantic Search in Millions of Equations*, page 135–143. Association for Computing Machinery, New York, NY, USA.
- José María González Pinto, Simon Barthel, and Wolf-Tilo Balke. 2014. Qualibeta at the ntcir-11 math 2 task: An attempt to query math collections. In *NTCIR-11*.
- EJG Pitman. 1937a. Significance tests which may be applied to samples from any populations. *Journal of the Royal Statistical Society*, B4.
- EJG Pitman. 1937b. Significance tests which may be applied to samples from any populations. ii. the correlation coefficient test. *Journal of the Royal Statistical Society*, B4.

- EJG Pitman. 1937c. Significance tests which may be applied to samples from any populations. iii. the analysis of variance test. *Biometrika*, 29.
- Minh Nghiem Quoc, Keisuke Yokoi, Yuichiroh Matsubayashi, and Akiko Aizawa. 2010. Mining coreference relations between formulas and text using wikipedia.
- Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. <http://is.muni.cz/publication/884893/en>.
- Marek Rei and Helen Yannakoudakis. 2016. Compositional Sequence Labeling Models for Error Detection in Learner Writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.
- Marek Rei, Gamal K. O. Crichton, and Sampo Pyysalo. 2016. Attending to Characters in Neural Sequence Labeling Models. In *Coling 2016*.
- Jean-Marc Richard. 2004. Safe domain and elementary geometry. *European Journal of Physics*, 25(6):835.
- Anna Ritchie, Simone Teufel, and Stephen Robertson. 2006. Creating a test collection for citation-based ir experiments. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 391–398, New York City, USA, June. Association for Computational Linguistics.
- S. E. Robertson and S. Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '94*, pages 232–241, New York, NY, USA. Springer-Verlag New York, Inc.
- S. E. Robertson, C. J. van Rijsbergen, and M. F. Porter. 1981. Probabilistic models of indexing and searching. In *Proceedings of the 3rd Annual ACM Conference on Research and Development in Information Retrieval, SIGIR '80*, pages 35–56, Kent, UK, UK. Butterworth & Co.
- Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at trec-3. In Donna K. Harman, editor, *TREC*, volume Special Publication 500-225, pages 109–126. National Institute of Standards and Technology (NIST).
- S. E. Robertson. 1977. The probability ranking principle in ir. *The Journal of Documentation*, 33:294–304.
- David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323:533–536.
- Michal Ruzicka, Petr Sojka, and Martin Líska. 2014. Math indexer and searcher under the hood: History and development of a winning strategy. In *NTCIR*.
- Tetsuya Sakai, Daisuke Ishikawa, and Noriko Kando. 2010. Overview of the ntcir-8 community qa pilot task (part ii): System evaluation. In *NTCIR*.

- G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November.
- G. Salton. 1971. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
- Tefko Saracevic. 1975. Relevance: A review of and a framework for thinking on the notion in information science. In *Journal of the American Society of Information Science*, volume 26, pages 321–243.
- Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.
- Linda Schamber, Michael Eisenberg, and Michael S. Nilan. 1990. A re-examination of relevance: Toward a dynamic, situational definition. *Inf. Process. Manage.*, 26(6):755–776, November.
- Linda Schamber. 1994. Relevance and information behavior. *Annual Review of Information Science and Technology (ARIST)*, 29:3–48.
- Henry Scheffe. 1943. Statistical inference in the non-parametric case. *Ann. Math. Statist.*, 14(4):305–332, 12.
- Thomas Schellenberg, Bo Yuan, and Richard Zanibbi. 2012. Layout-based substitution tree indexing and retrieval for mathematical expressions. pages 82970I–82970I–8.
- Moritz Schubotz, Alexey Grigorev, Marcus Leich, Howard S. Cohl, Norman Meuschke, Bela Gipp, Abdou S. Youssef, and Volker Markl. 2016. Semantification of identifiers in mathematics for better math information retrieval. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '16*, pages 135–144, New York, NY, USA. ACM.
- Ruocheng Shan and Abdou Youssef. 2021. Towards math terms disambiguation using machine learning. In Fairouz Kamareddine and Claudio Sacerdoti Coen, editors, *Intelligent Computer Mathematics*, pages 90–106, Cham. Springer International Publishing.
- J Sharp. 1964. Review of the cranfield-wru test literature. *Journal of Documentation*, 20:55–69.
- S. Siegel and N.J. Castellan. 1988. *Nonparametric statistics for the behavioral sciences*. McGraw-Hill, Inc., second edition.
- Mark D. Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM '07*, pages 623–632, New York, NY, USA. ACM.
- Ian Soboroff. 2014. Computing confidence intervals for common IR measures. In Stefano Mizzaro and Ruihua Song, editors, *Proceedings of the Sixth International Workshop on Evaluating Information Access, EVIA 2014, National Center of Sciences, Tokyo, Japan, December 9, 2014*. National Institute of Informatics (NII).
- K. Spärck-Jones and C.J. van Rijsbergen. 1975. Report on the first stage of an investigation into the comparative efficiency of indexing systems. Technical report, Cambridge University Computer Laboratory, Cambridge.

- K. Spärck-Jones and C.J. van Rijsbergen. 1976. Information retrieval test collections. *Journal of Documentation*, 32(1):59–75.
- Karen Spärck-Jones and Peter Willett, editors. 1997. *Readings in Information Retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Karen Spärck-Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21.
- Karen Spärck-Jones. 1981. *Information retrieval experiment*. Butterworth.
- Heinrich Stamerjohanns and Michael Kohlhase. 2008. Transforming the arxiv to xml. In Serge Autexier, John Campbell, Julio Rubio, Volker Sorge, Masakazu Suzuki, and Freek Wiedijk, editors, *Intelligent Computer Mathematics*, volume 5144 of *Lecture Notes in Computer Science*, pages 574–582. Springer Berlin Heidelberg.
- Heinrich Stamerjohanns and Michael Kohlhase. 2009. arXMLiv: Translating the Math Archives to XML+MathML.
- Heinrich Stamerjohanns, Michael Kohlhase, Deyan Ginev, Catalin David, and Bruce R. Miller. 2010. Transforming large collections of scientific publications to xml. *Mathematics in Computer Science*, 3(3):299–307.
- Yiannos Stathopoulos and Simone Teufel. 2015. Retrieval of research-level mathematical information needs: A test collection and technical terminology experiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 334–340.
- Yiannos Stathopoulos and Simone Teufel. 2016. Mathematical information retrieval based on type embeddings and query expansion. In *Proceedings of the 26th International Conference on Computational Linguistics, Coling 2016, December 11-16, 2016, Osaka, Japan*, pages 334–340.
- Yiannos Stathopoulos, Simon Baker, Marek Rei, and Simone Teufel. 2018. Variable typing: Assigning meaning to variables in mathematical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 303–312, New Orleans, Louisiana, June. Association for Computational Linguistics.
- Don R. Swanson. 1965. The evidence underlying the cranfield results. *The Library Quarterly*, 35(1):pp. 1–20.
- Mortimer Taube and Harold Wooster. 1958. *Information storage and retrieval : theory, systems and devices / ed. by Mortimer Taube and Harold Wooster : [bibliogs.]*. Columbia Univ. Press New York.
- Yla R. Tausczik and James W. Pennebaker. 2011. Predicting the perceived quality of online mathematics contributions from users’ reputations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ’11*, pages 1885–1888, New York, NY, USA. ACM.

- Yla R. Tausczik and James W. Pennebaker. 2012. Participation in an online mathematics community: Differentiating motivations to add. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, CSCW '12*, pages 207–216, New York, NY, USA. ACM.
- Yla R. Tausczik, Aniket Kittur, and Robert E. Kraut. 2014. Collaborative problem solving: A study of mathoverflow. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing, CSCW '14*, pages 355–367, New York, NY, USA. ACM.
- Jaime Teevan, Christine Alvarado, Mark S. Ackerman, and David R. Karger. 2004. The perfect search engine is not enough: A study of orienteering behavior in directed search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '04*, pages 415–422, New York, NY, USA. ACM.
- Goran Topic, Giovanni Yoko Kristianto, Minh-Quoc Nghiem, and Akiko Aizawa. 2013. The MCAT math retrieval system for NTCIR-10 math track. In *Proceedings of the 10th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-10, National Center of Sciences, Tokyo, Japan, June 18-21, 2013*.
- Vladimir Vapnik. 1982. *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Springer-Verlag, Berlin, Heidelberg.
- B. C. Vickery. 1967. Reviews of cleverdon, c. w., mills, j. and keen, e. m. the cranfield 2 report. *Journal of Documentation*, 22:247–249.
- Ellen M. Voorhees and Donna K. Harman. 2005. *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press.
- Ellen M. Voorhees. 1998. Variations in relevance judgments and the measurement of retrieval effectiveness. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '98*, pages 315–323.
- E. Voorhees. 2002. The philosophy of information retrieval evaluation. In *Evaluation of cross-language information retrieval systems*, pages 143–170. Springer.
- P.J. Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.
- Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, December.
- Magdalena Wolska and Mihai Grigore. 2010. Symbol declarations in mathematical writing. In *Towards a Digital Mathematics Library. Paris, France, July 7-8th, 2010*, pages 119–127. Masaryk University Press.
- Magdalena Wolska, Mihai Grigore, and Michael Kohlhase. 2011. Using discourse context to interpret object-denoting mathematical expressions. In *Towards a Digital Mathematics Library. Bertinoro, Italy, July 20-21st, 2011*, pages 85–101. Masaryk University Press.

- David Yenicelik, Florian Schmidt, and Yannic Kilcher. 2020. How does BERT capture semantics? a closer look at polysemous words. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 156–162, Online, November. Association for Computational Linguistics.
- Keisuke Yokoi and Akiko Aizawa. 2009. An approach to similarity search for mathematical expressions using mathml. In *Towards a Digital Mathematics Library. Grand Bend, Ontario, Canada, July 8-9th, 2009*. Masaryk University Press.
- Abdou Youssef and Bruce R. Miller. 2019. Explorations into the use of word embedding in math search and math semantics. In Cezary Kaliszyk, Edwin Brady, Andrea Kohlhase, and Claudio Sacerdoti Coen, editors, *Intelligent Computer Mathematics*, pages 291–305, Cham. Springer International Publishing.
- Abdou S. Youssef. 2007. Methods of relevance ranking and hit-content generation in math search. In Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *Towards Mechanized Mathematical Assistants*, pages 393–406, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. 2007. A support vector method for optimizing average precision. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '07*, page 271–278, New York, NY, USA. Association for Computing Machinery.
- Richard Zanibbi and Bo Yuan. 2011. Keyword and image-based retrieval of mathematical expressions. In Agam and Viard-Gaudin (Schellenberg et al., 2012), pages 1–10.
- Richard Zanibbi, Douglas W. Oard, Anurag Agarwal, and Behrooz Mansouri. 2020. Overview of arqmath 2020: Clef lab on answer retrieval for questions on math. In Avi Arampatzis, Evangelos Kanoulas, Theodora Tsirikia, Stefanos Vrochidis, Hideo Joho, Christina Lioma, Carsten Eickhoff, Aurélie Névéol, Linda Cappellato, and Nicola Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 169–193, Cham. Springer International Publishing.
- Matthew D. Zeiler, Marc’Aurelio Ranzato, Rajat Monga, Mark Z. Mao, K. Yang, Quoc V. Le, Patrick Nguyen, Andrew W. Senior, Vincent Vanhoucke, Jeffrey Dean, and Geoffrey E. Hinton. 2013. On rectified linear units for speech processing. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3517–3521.
- Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*.
- Chengxiang Zhai and John Lafferty. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '01*, pages 334–342, New York, NY, USA. ACM.
- Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, April.

Qun Zhang and Abdou Youssef. 2014. An approach to math-similarity search. In Stephen M. Watt, James H. Davenport, Alan P. Sexton, Petr Sojka, and Josef Urban, editors, *Intelligent Computer Mathematics*, pages 404–418, Cham. Springer International Publishing.

Justin Zobel. 1998. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '98*, pages 307–314, New York, NY, USA. ACM.

# Appendix A

## Technical Background

### A.1 Natural Language Processing

In this section I will present a brief overview of NLP methods relevant to this work. Many aspects of my work rely on text tokenisation (Section A.1.1), part-of-speech tagging (Section A.1.2) and dependency parsing (Section A.1.3) as pre-processing steps. I used the Stanford CoreNLP toolkit (Manning et al., 2014) to perform these tasks.

#### A.1.1 Word Tokenisation and Sentence Segmentation

Tokenisation is the task of grouping streams of characters into meaningful units of information (Manning et al., 2008; Jurafsky and Martin, 2008). In IR tokenising text into its constituent words is an important pre-processing step for constructing an inverted index (Manning et al., 2008).

Word tokenisation is non-trivial and cannot be performed effectively by splitting text at white spaces and punctuation marks (Manning et al., 2008; Jurafsky and Martin, 2008). Word tokenisers must take into consideration abbreviations, hyphenated words, numbers and phenomena involving apostrophes, such as contractions and genitive markers. Thus, modern word tokenisers take the form of binary character classifiers that are trained using machine learning (Jurafsky and Martin, 2008).

Segmentation of text into sentences is also non-trivial. Sentence boundaries, such as start-of-sentence capitalisation and end-of-sentence punctuation (e.g., “.”, “?”, and “!”), often occur inside word tokens such as numbers, abbreviations and named entities (Jurafsky and Martin, 2008). As a result, modern sentence tokenisers take the form of binary character classifiers making decisions as to whether a certain character is a sentence boundary marker (Jurafsky and Martin, 2008).

### A.1.2 Part-of-Speech Tagging

Part-of-speech (POS) tagging is the task of labelling each word in a corpus with its part of speech: a grammatical category representing words that behave similarly syntactically. For example, words can be nouns, verbs, pronouns, prepositions, adverbs, conjunctions, particles and articles (Jurafsky and Martin, 2008).

POS tagging is useful in NLP because it enriches individual word tokens with additional information. POS tagging informs an NLP system on how each token relates to its neighbours (the context), gives limited semantic information about the token and provides information on how tokens fit together to form a sentence (Jurafsky and Martin, 2008).

### A.1.3 Dependency Parsing

Dependency parsing is a form of syntactic parsing based on the dependency grammar formalism. Syntactical structure is captured by binary, directed relations between sentence tokens, rather than nested phrase structures. Each relation on a dependency tree connects a syntactically subordinate word (called the “dependent”) to its “head” word via a directed edge. Edges in a dependency tree are labelled by the grammatical function of the relation. For example, verb–subject, verb–object and noun–modifier relations are labelled using distinct labels on a dependency tree.

Dependency parsing is useful in many NLP tasks that rely on extracting machine learning features based on the syntactic structure of sentences, such as machine translation and information extraction. Nivre (2010) attributes this to two advantages of dependency representations. First, dependency structures are efficient to produce and can be constructed in linear time. Second, dependency representations make token-level features directly available to downstream tasks.

### A.1.4 Word Embeddings

In a large corpus with vocabulary  $V$  a word  $w$  can be represented by a *one-hot* vector: a vector with  $|V|$  components all set to zero except the component corresponding to  $w$ , which is set to 1. The components of a one-hot vector are orthogonal (i.e., independent) and as a result, one-hot vectors do not capture the relationship between words in a particular context (e.g., sentence, paragraph, document, corpus). Furthermore, one-hot vectors grow larger as the vocabulary and corpus become bigger.

Word embeddings (or embeddings for short) are vector representations of words in a continuous vector space: each word becomes an  $n$ -dimensional vector with real-valued components. These vectors are called embeddings because they are projections of large one-hot vectors into a denser, continuous space.

In this work I used the skipgram, `Word2Vec` model (Mikolov et al., 2013b) to construct embeddings for types in the MREC (cf. Section 4.3). The skip-gram model is a neural language model that predicts the context of a word: it learns the probability distribution of the words most likely to appear near the input word within a window of size  $C$ . Effectively, the skip-gram model

embeds word vectors into a dense, continuous space that captures the relationship of each word to the contexts in which it appears.

To produce training data for the skip-gram model, the model slides a fixed-length window of size  $2 \times C$  over every sentence in the corpus. The word in the centre of the window is referred to as the *target word* and is neighbored by  $C$  words to its left and right. For example, consider the sentence:

*I will have a sandwich for lunch.*

With target word “sandwich” and  $C = 2$ , the words neighbouring the target word are “will”, “have”, “for”, “lunch”, as shown in Figure A.1, where the target word is in blue and its neighbours are colored with light blue.



Figure A.1: Example sliding window of size 2: target word and its neighbouring words.

The iteration of the sliding window method shown in Figure A.1 will generate the training data  $\{(sandwich, will), (sandwich, have), (sandwich, for), (sandwich, lunch)\}$ . In the next iteration of the method, the target word becomes “for” and more pairs are generated with the process repeating until all words in all sentences of the training corpus are iterated over.

Pairs generated for each target word are used to train a fully connected multi-layer perceptron (MLP)<sup>1</sup> that predicts the context (the words likely to be in the window) of the target word.

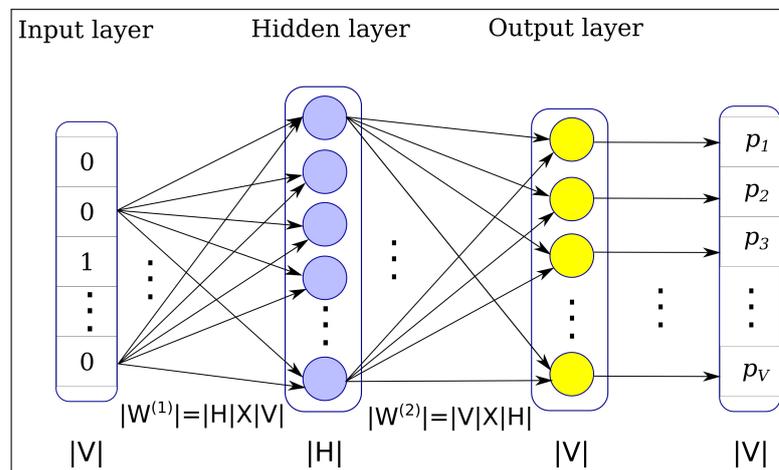


Figure A.2: MLP architecture for training the skip-gram model.

The architecture of the MLP used to train the `word2vec` skip-gram model is shown in Figure A.2. The task of training a skip-gram model is setup as follows.

<sup>1</sup>Described Section A.2.2

The goal is to train the MLP to predict the probability that each word in the vocabulary will be part of the context window of the input target word. This is a multi-class classification task with as many classes as there are words in the vocabulary. The input to the MLP is a one-hot vector representation for the target word of size  $|V| \times 1$ , where  $|V|$  is the size of the vocabulary, as shown in Figure A.2.

The size  $|H|$  of the hidden layer is a hyper-parameter of the model that determines the dimensionality of the word vector representations. The neurons on the hidden layer (blue circles in Figure A.2) have no activation function, so:

$$\mathbf{f}^{(1)} = \mathbf{z}^{(1)} = \mathbf{w}^{(1)}\mathbf{x}.$$

In other words, the hidden layer computes a linear projection of the input and the shape of the weight matrix for the hidden layer is  $|W| = |H| \times |V|$ , since the hidden layer is fully connected.

The neurons on the output layer (yellow circles in Figure A.2) use the softmax function (Section A.2.2.2) as their activation and the weight matrix  $\mathbf{W}^{(2)}$  for the output layer has shape  $|V| \times |H|$ .

The feed forward step is described as follows. For each target word in a sentence, the one-hot representations of its context words are input to the MLP ( $2 \times C$  one-hot vectors) and the hidden layer computes the dot product of  $\mathbf{W}^{(1)}$  and the input vector. Since the input vector is a one-hot vector with all but one of its components having the value 0, the dot product between the input and the weight matrix  $\mathbf{W}^{(1)}$  will have the effect of selecting the vector representation of the input word from the weight matrix (the  $i^{\text{th}}$  row of  $\mathbf{W}^{(1)}$ , where  $w$  is the component of the input set to 1). Thus, the output of the hidden layer is a  $1 \times |H|$  word vector for the input word, which is fedforward to the output layer.

The output of each neuron on the output layer is normalised to a probability distribution using softmax activation (Section A.2.2.2). A prediction vector for each input target word is obtained from the softmax activation function, which is used to normalise the output of each neuron on the output layer into a probability. Training is performed using backpropagation with gradient descent<sup>2</sup> and cross-entropy<sup>3</sup> as the cost function. Once training is complete over all sentences and contexts, embeddings for the words in the corpus can be extracted as the columns from the weight matrix of the output layer,  $\mathbf{W}^{(2)}$ .

---

<sup>2</sup>discussed in Section A.2.2

<sup>3</sup>discussed in Section A.2.2.1

### A.1.5 The C-value/NC-value algorithm

One of the most commonly used algorithms for technical terminology detection is the C-Value algorithm (Frantzi et al., 1998). The algorithm gathers corpus-wide frequencies about a potential multi-word term, its subsequences and all sequences that contain the potential term as subsequences. The score for the C-Value component is computed as:

$$C - Value(a) = \begin{cases} \log_2(|a|) \times f(a) & \text{if } a \text{ is not nested} \\ \log_2(|a|) \times (f(a) - \frac{1}{P(T_a)} \sum_{b \in T_a} f(b)) & \text{otherwise} \end{cases}$$

where  $a$  is the candidate string,  $f(a)$  is the corpus frequency of  $a$ ,  $T_a$  is the set of extracted candidate terms that contain  $a$  and  $P(T_a)$  is the cardinality of  $T_a$ . The algorithm also applies linguistic filters in the form of a stop-word list and regular expressions over part-of-speech sequences.

The NC-Value component of the algorithm, which re-ranks the ranks provided by the C-Value score, harvests contextual information in the form of words (nouns, adjectives and verbs) from a text window surrounding a candidate. Context words are selected if they appear in the context of many technical terms. This criterion is formalised using the measure:

$$weight(w) = \frac{t(w)}{n}$$

where  $w$  is the context word (noun, verb or adjective) to be weighted as a potential technical term context word,  $t(w)$  is the number of technical terms  $w$  appears with and  $n$  the total number of technical terms considered.

Reranking proceeds for each candidate technical term by computing the context factor by summing up  $f_a(b) \times weight(b)$  for each  $b$  in the context of the candidate term. The final score for a candidate term  $a$  is given by:

$$score(a) = 0.8C - Value(a) + 0.2 \sum_{b \in C_a} f_a(b) \times weight(b)$$

where  $C_a$  is the set of distinct context words for  $a$ ,  $b$  is a word from  $C_a$  and  $weight(b)$  is the weight of the technical term context word  $b$  as described earlier.

## A.2 Machine Learning

In this section, I will introduce machine learning concepts that are prerequisites to my presentation of the variable typing task in chapter 5.

Machine learning (ML) methods solve computational problems that cannot be easily solved by programs written by humans. Input to ML algorithms is a *data set of examples*. An example is a set of *features* – quantitative measures about the object or processes being modelled by

the ML algorithm. ML algorithms process each example and construct a *model* of the data – a function that describes or summarises the entire set of examples. *Unsupervised learning* methods process examples to produce a function that identifies certain properties of the data set, such as a probability distribution. *Supervised learning* methods model the data by constructing a map from input features to a target *label* or class that has been assigned to each example by a human instructor.

Once trained, useful ML models *generalise to unseen* or *test* examples. A model that generalises well makes good *predictions* for examples not observed during training. To measure the generalisability of a model, its performance with respect to some measure  $M$ , is quantified on a *test set* of examples collected separately from the training data.

Most ML algorithms have *hyperparameters* that control the behaviour of the learning algorithm. The values of the hyperparameters are not optimised during the learning process. Instead, a small sub-set of the training data, referred to as the *validation* or *development* set is used to determine good values for the hyperparameters of the learning algorithm. An alternative to creating three separate data sets for training, validation and testing is cross validation.

In cross validation, the data set is split into  $n$  partitions, or folds, where typically  $n = 10$ . Training and evaluation with cross validation is repeated  $n - 1$  times and at each iteration one partition is reserved for testing while the remaining partitions are used for training and validation. This allows a machine learning model to be trained and evaluated effectively when not much data is available. Specifically, the entire data set can be used for training and evaluation and at any iteration data points used for training are not also used for evaluation. Model performance and training error can be summarised by taking the average of these metrics across  $n$  folds.

A machine learning algorithm performs well when the training error is minimised and the difference between training and test error is small. *Underfitting* occurs when a model is not able to fit the training data well (i.e., training error is large). *Overfitting* occurs when the gap between the training and test error is too large – the model has “memorised” the training data.

The machine learning methods I will discuss in this section are instances of *supervised learning* – they make use of labelled data to build models that generalise to unseen data.

### **A.2.1 Support Vector Machines**

Support Vector Machines (SVM) are supervised learning classifiers that separate examples into classes by estimating the optimal decision boundaries (Vapnik, 1982; Cortes and Vapnik, 1995). In the linear binary classification case, the decision boundary between two classes is an  $m$ -dimensional hyperplane that separates the points into two classes such that the distance between classes in the hyperspace is maximised. The SVM in Figure A.3, for example, has ten training examples, each defined by two features (feature 1 and feature 2) and labelled as either positive or negative.

The problem of finding the optimal decision hyperplane is equivalent to finding the subsets of positive and negative training examples that lie closest to the decision boundary, referred to as

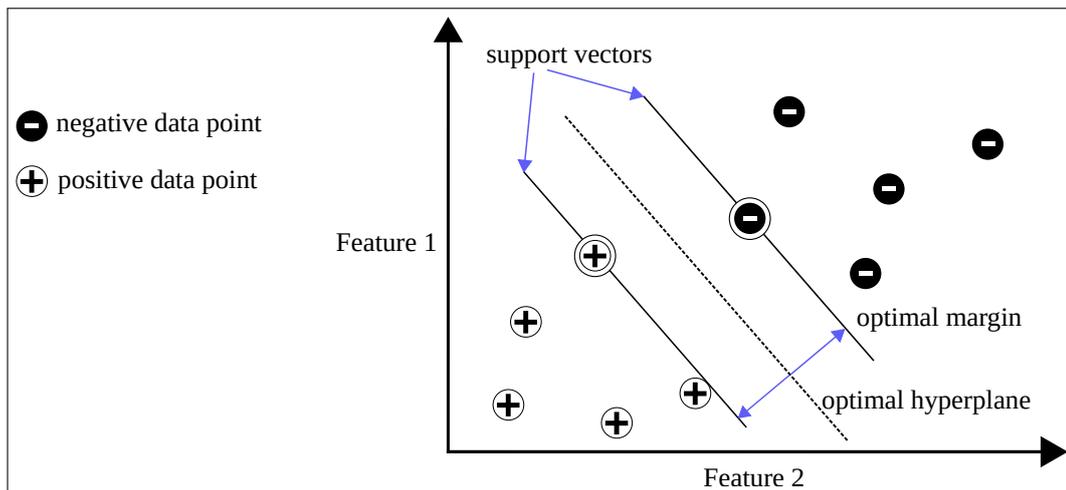


Figure A.3: Illustration of Support vector machines.

support vectors. The support vectors are mathematically defined to be:

$$H_+ : w_0^T \mathbf{x} + b = 1$$

$$H_- : w_0^T \mathbf{x} + b = -1$$

where  $w_0$  is the matrix feature weights to be learned during training (one component per feature) and  $x$  is the example feature matrix. The decision boundary is the hyperplane  $H_0$  that is the median of the support vectors:

$$H_0 : w_0^T \mathbf{x} + b = 0$$

Training of SVMs can be conceptualised as adjusting the support vector weights such as the optimal margin (as shown in Figure A.3) is maximised. Numerically, this is done using constraint satisfaction.

Cortes and Vapnik (1995) introduced soft-margin SVMs that optimise the margin between classes while also allowing for misclassified examples. This enables linear SVMs to be flexible with non-linearly separable data sets and to learn how to classify noisy data with many outlier examples.

Soft-margin SVMs have a hyperparameter  $C$ , which controls how much the SVM optimisation process will avoid misclassifying training examples. Large values of  $C$  make the optimization process choose a smaller-margin hyperplane, minimising misclassifications while very small values of  $C$  will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

In some cases, examples are not linearly separable – there are too many misclassifications for the SVM to be useful in making predictions. SVMs can use the *kernel trick* to solve non-linear problems using linear optimisation techniques (Boser et al., 1992). A *kernel* is a function that transforms the raw feature space (i.e., the space defined by the raw features) into a hyperspace that produces fewer misclassifications when the same linear separation methods are used to

estimate the optimal decision hyperplane.

In this thesis, I use binary linear SVM classification models for my variable typing machine learning task (Chapter 5).

### A.2.2 Neural Networks

I will introduce the topic of neural networks starting with feedforward neural networks, also known as multilayer perceptrons (MLPs), because they are foundational to the topic and are conceptual prerequisites to the neural networks architectures I will introduce later in this section.

The goal of an MLP is to learn a function  $y = g(x)$  by approximating it with a mapping  $\mathbf{y} = \hat{g}(\mathbf{x}; \theta)$ , where  $\theta$  are the parameters of the model (much like the coefficient and intercept in a linear regression model) and  $\mathbf{x}$ ,  $\hat{\mathbf{y}}$  are the input and output vectors of the model, respectively.

In MLPs the approximation  $\hat{g}$  to  $g$  takes the form of a chaining of functions:

$$\hat{g}(\mathbf{x}) = f^{(n)}(f^{(n-1)}(\dots f^{(1)}(x))) \quad (8)$$

Each function  $f^{(l)}$  is a function computed by the layers of the neural network as shown in Figure A.4

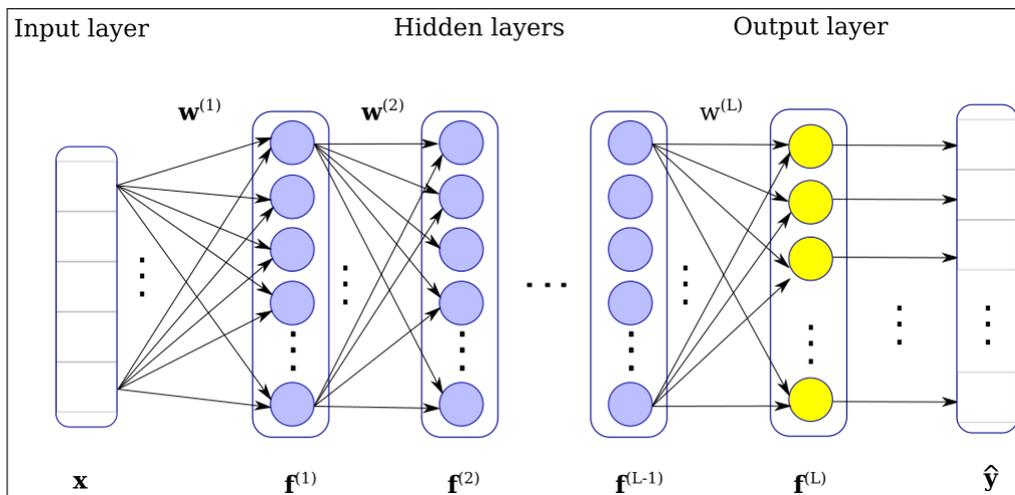


Figure A.4: A fully connected multilayer perceptron (MLP).

In fact, Figure A.4 shows the general form of a feedforward neural network:

- it takes a vector  $\mathbf{x}$  as input and has an output vector  $\hat{\mathbf{y}}$ ;
- it has  $L$  layers with  $L - 1$  *hidden layers* – layers whose influence to the output is indirect;
- it is a fully connected MLP, with the output of every neuron (shown as circles, blue for hidden layers and yellow for the output layer in Figure A.4) at layer  $l$  becoming the input to every neuron of layer  $(l + 1)$ ;

- the neurons at layer  $l$  have *activation function*  $\sigma^{(l)}$  that determines which neurons will produce non-zero output based on the values the neurons receive as input from the previous layer;
- every layer  $l$  has a weight matrix  $\mathbf{w}^{(l)}$  of dimensions  $|f^{(l)}| \times |f^{(l-1)}|$ , where  $|f^{(l)}|$  is the number of neurons at layer  $l$ .

The output of each layer is a function of the output of the previous layer as defined by the recurrence relations:

$$f^{(1)} = \sigma^{(1)}(\mathbf{w}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) \quad (9)$$

$$f^{(l+1)} = \sigma^{(l+1)}(\mathbf{w}^{(l+1)}f^{(l)} + \mathbf{b}^{(l+1)}) \quad (10)$$

The weights and biases of the MLP are initialised at random. Then, a prediction for each example can be obtained by going through a feedforward step: the input vector is applied to the function chain (equation 8) by the neural network through the recursive application of equations 9 and 10 to produce the output vector.

Learning in neural networks involves minimising the prediction error, determined by a cost function  $C$ , over all training instances by adjusting the parameters of the model. This is an optimisation problem with the goal of minimising the cost function with respect to the weights and biases of the neural network. One technique used to solve this optimisation problem is *gradient descent* and the idea behind it is as follows.

Cost is a function of the parameters of the model in a high dimensional space (there can be thousands of parameters). By computing its gradient with respect to weights and biases we obtain a vector in the space defined by the possible inputs (weights and biases included) and outputs (the cost range) of the cost function. Intuitively, the components of the gradient vector tell us the direction and magnitude each parameter (component in the gradient vector) needs to be changed so that the overall cost is reduced. Gradient descent iteratively reduces the cost function by changing the parameters according to the gradient until a local or global minimum for the cost function is reached.

Estimating the mean error of the network over all training instances is computationally expensive, particularly when there are thousands of training examples. To solve this problem a technique called *mini-batching* is used. In mini-batching the set of training examples is divided into randomly sampled batches. The examples in each batch are fedforward and the cumulative error of the network is computed on the batch so that the parameters of the model can be updated accordingly.

Learning with gradient descent requires that the training process computes the gradient of the cost function with respect to the parameters of the neural network. This computation is done using the *backpropagation algorithm* (Rumelhart et al., 1986).

Let us first discuss the intuitions behind how backpropagation computes the desired gradient. A change in a particular weight (or bias) in the network will also cause a change in the output activation of the corresponding neuron. In turn, this will cause change in all activations in the next layer<sup>4</sup> and all subsequent layers, ultimately changing the output of the network and the cost function by some value.

Thus, every edge from the first weight to its corresponding neuron has some rate of change associated with it. Therefore, to calculate the rate of change of the cost function with respect to a particular weight we sum up the rates of change of all possible paths from the weight to the final cost.

As discussed in previous paragraphs, the output of the neural network is a composition of functions which depend on the weights and biases of the network. Backpropagation uses the chain rule to compute the gradient and propagates partial derivatives of the cost backwards, by computing the partial derivative for layer  $l$  using that of layer  $l + 1$ , starting from the output layer.

These intuitions are formalised by the four equations of backpropagation, presented below in matrix form<sup>5</sup>:

$$\delta^{(L)} = \nabla_f C \odot \sigma'(\mathbf{z}^{(L)}) \quad (\text{i})$$

$$\delta^{(l)} = ((\mathbf{w}^{(l+1)})^T \delta^{(l+1)}) \odot \sigma'(\mathbf{z}^{(l)}) \quad (\text{ii})$$

$$\frac{\delta C}{\delta b_j^l} = \delta_j^l \quad (\text{iii})$$

$$\frac{\delta C}{\delta w_{j,k}^{(l)}} = f_k^{(l-1)} \delta_j^{(l)} \quad (\text{iv})$$

where

$$\nabla_f C = \frac{dC}{df^{(L)}} = C'(\mathbf{f}^{(L)} - (y)),$$

$\mathbf{f}^{(l)} = \sigma(\mathbf{z}^{(l)})$  is the vector of activations

for the neurons of layer  $l$ ,

$f^{(l)}$  is the activation vector for layer  $l$ ,

$\mathbf{z}^{(l)} = \mathbf{w}^{(l)}\mathbf{f}^{(l-1)} + \mathbf{b}$ , a helper function denoting the *weighted input* to layer  $l$

$\delta^{(l)}$  is a vector representing the rate of change of the cost wrt  $\mathbf{z}^{(l)}$  (error) for layer  $l$ ,

$f_k^l$  is the activation of neuron  $k$  at layer  $l$ ,

$\odot$  is the Hadamard product: element-wise multiplication of two vectors,

of equal size.

<sup>4</sup>Assuming a dense (fully connected) network

<sup>5</sup>Boldface symbols represent matrices or vectors.

Equations (1) and (2) give rate of change of the cost function wrt  $\mathbf{z}$  at the output layer and hidden layer  $l$ , respectively. Equation (3) gives the rate of change of the cost function wrt the bias for neuron  $j$  of layer  $l$  and equation (iv) gives the gradient of the cost function wrt the weight from neuron  $j$  of layer  $l$  to neuron  $k$  of layer  $l - 1$ .

Gradient descent updates a particular weight or bias of the neural network by subtracting its computed gradient (equations (iii) and (iv), respectively) multiplied by the *learning rate*,  $\eta$ . The learning rate is a hyperparameter of the model that controls how rapidly adjustments to the parameters traverse the parameter space of the cost function to converge to the local or a global minimum.

We now have all the necessary components to write down the algorithm for training neural networks with gradient descent, mini-batching and backpropagation. The algorithm is shown in Algorithm 1.

---

**Algorithm 1** Gradient Descent with Minibatching and Backpropagation

---

**Require:** A set of training examples,  $X$

**Require:** Number of epochs,  $E$

**Require:** A learning rate  $\eta$

$M \leftarrow$  split  $X$  into  $|M|$  mini-matches

**while**  $E \neq 0$  **do**

**for each** mini-batch  $m \in M$  **do**

**for each** training example  $x \in m$  **do**

**for each** layer  $l \in \{1, \dots, L\}$  **do** ▷ Feedforward step

$$\mathbf{z}^{x,(l)} = \mathbf{w}^{(l)} \times \mathbf{f}^{x,(l-1)} + \mathbf{b}^{(l)}$$

$$\mathbf{f}^{x,(l)} = \sigma(\mathbf{z}^{x,(l)})$$

**end for**

$$\delta^{x,(L)} = \nabla_f C_x \odot \sigma'(\mathbf{z}^{x,(L)})$$
 ▷ Compute output error

**for each**  $l \in \{L - 1, \dots, 1\}$  **do** ▷ Backpropagate the error

$$\delta^{x,(l)} = ((\mathbf{w}^{(l+1)})^T \delta^{x,(l+1)}) \odot \sigma'(\mathbf{z}^{x,(l)})$$

**end for**

**end for**

**for each**  $l \in \{L, L - 1, \dots, 1\}$  **do** ▷ Gradient Descent

$$\mathbf{w}^{(l)} \leftarrow \mathbf{w}^{(l)} - \frac{\eta}{|m|} \sum_x \delta^{x,(l)} (\mathbf{f}^{x,(l-1)})^T$$
 ▷ Update the weights

$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \frac{\eta}{|m|} \sum_x \mathbf{b}^{x,(l)}$$
 ▷ Update the biases

**end for**

**end for**

$E \leftarrow E - 1$

**end while**

---

A few notes on Algorithm 1:

1. An *epoch* is a complete training pass over all training examples.
2. I have omitted additional termination conditions. For example, it is sensible to stop early (i.e., before the algorithm completes the specified number of epochs) when the parameters remain unchanged (a minimum has been reached) or change very little.
3. The additional superscript  $x$  in some terms (e.g.,  $\delta^{x,(l)}$ ) indicates that the value is remembered or retrieved using  $x$  to index the term's value.
4. The weight and bias updates in Algorithm 1 are presented in matrix form, rather than in the component form of equations (iii) and (iv).

In practice activation functions for the neurons at each layer, the cost function and gradient optimisation technique are selected based on the task at hand and I will briefly introduce a few important ones.

### A.2.2.1 Cost Functions

Cost functions measure learning performance and the worse the predictions of a neural network become by changing its parameters, the higher the value of the cost function.

**Mean Square Error** or MSE is the mean of the sum of squares of the differences between true ( $y$ ) and predicted ( $\hat{y}$ ) values:

$$MSE(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^N (y - \hat{y})^2$$

MSE is most commonly used as the cost function in regression.

**Cross-Entropy Cost function** measures the performance of models making predictions about the probability of membership to discrete classes and is thus commonly used with classification tasks. The cross-entropy cost function is defined to be:

$$Cross - Entropy = - \sum_{c \in C} p(c) \times \log q(c)$$

where  $p(c)$  is the probability of membership in class  $c$  predicted by the model and  $q$  the true probability of  $c$ . The formula above is for multi-class classification (more than two discrete classes). In the case of binary classification the Cross-Entropy formula can be unrolled to produce the formula for binary Cross-Entropy cost:

$$Binary Cross - Entropy = -[p(c_0) \times \log(q(c_0)) + P(c_1) \times \log(q(c_1))]$$

### A.2.2.2 Activation Functions

Recall that an activation function  $\sigma$  is applied to the weighted input of layer  $l$  to produce its activation output  $\mathbf{f}^{(l)}$ :

$$\sigma(\mathbf{z}^{(l)}) = \sigma(\mathbf{w}^{(l)}\mathbf{f}^{(l-1)} + \mathbf{b})$$

In the discussions that follow I will continue to denote the weighted input to  $\sigma$  as  $z$ .

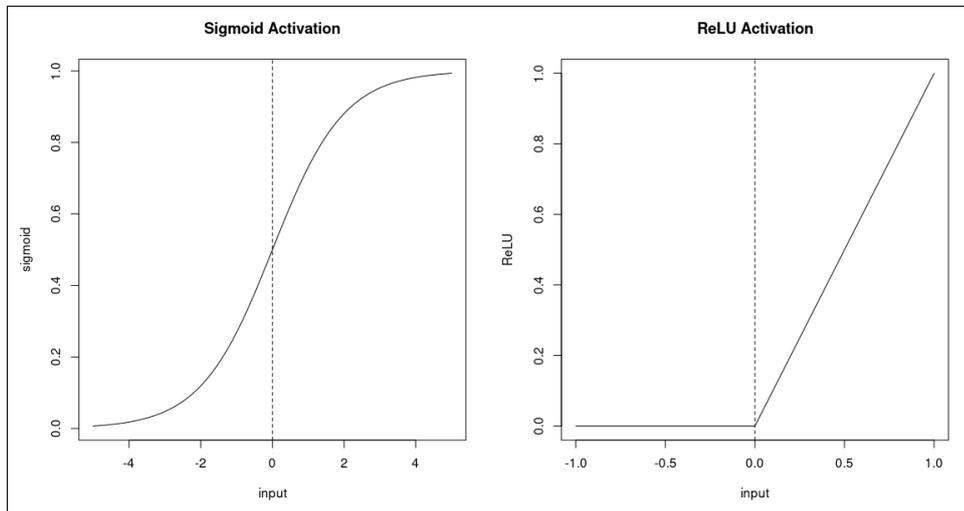


Figure A.5: The Sigmoid (left) and ReLU (right) activation functions.

**Sigmoid** activation function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

As shown in Figure A.5 (left), it is an S-shaped curve that is convex for negative inputs and concave for positive inputs, with values bounded between 0 and 1. It is a differentiable, real-valued and monotonic function with a non-monotonic derivative. The sigmoid function is most often used with neural network layers that perform regression.

**ReLU** or *Rectified Linear Unit*, is defined by:

$$ReLU(z) = \max(0, z)$$

The range of values ReLU can output ranges from 0 to infinity. As shown in Figure A.5 (right), ReLU maps all negative inputs to 0 (hence the “rectified” in its name) and each positive input onto itself ( $ReLU(z) = z$ ). ReLU is differentiable and monotonic. Its derivative is also monotonic. The ReLU activation function introduces non-linearity in a neural network, making it suitable for use in classifiers where the classes to be predicted are characterised by non-linear features. For example, in image object detection with convolutional neural networks (Section A.2.2.4) ReLU is typically used as the activation function for the feature detection stage because objects

of interest contained in images are often non-linear. ReLU is understood to be an effective activation function for convolutional neural networks (Jarrett et al., 2009; Nair and Hinton, 2010; Glorot et al., 2011; Maas et al., 2013; Zeiler et al., 2013).

**tanh** or hyperbolic tangent, is another non-linear activation function defined by:

$$\tanh(z) = \frac{2}{1 + e^{-2z} - 1}$$

The tanh activation function is S-shaped, sharing its structure with the sigmoid function above. The tanh function is differentiable and its range is restricted to values between -1 and 1. Tanh is useful in recurrent neural networks to squash voting vectors back into the domain of the input.

**Softmax** is a multivariate activation function used in neural network architectures for multi-class classification. For a classification task with  $K$  classes ( $K > 2$ ), a softmax activation at the output layer with  $K$  neurons will output the probability for class  $i$  at neuron  $i$ . The softmax function is defined as:

$$\text{softmax}(z_i^{(l)}) = \frac{e^{z_i^{(l)}}}{\sum_j e^{z_j^{(l)}}}$$

Let us have a look at a short example. Figure A.6 shows a fully connected MLP with a softmax output layer. That is,  $f_i^{(2)} = \sigma(\text{softmax}(z_i^{(2)}))$ .

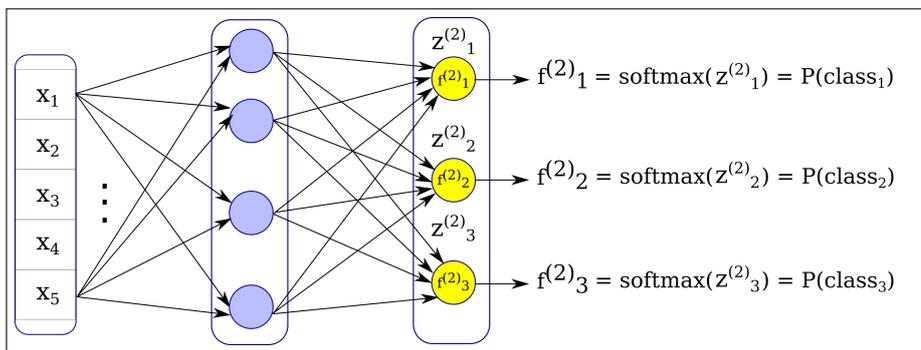


Figure A.6: Example of a fully-connected MLP for 3-class classification with a softmax activation layer.

The classifier in Figure A.6 takes a vector with 5 features as input, representing an observation, and decides the most probable of three classes for that observation. Suppose that the weighted inputs to the output layer of the network in Figure A.6 are:

$$\mathbf{z}^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} = \begin{bmatrix} 1.26 \\ 0.82 \\ 0.43 \end{bmatrix}$$

Then, the softmax activation output for the network will be

$$\mathbf{f}^{(2)} = \begin{bmatrix} f_1^{(2)} \\ f_2^{(2)} \\ f_3^{(2)} \end{bmatrix} = \begin{bmatrix} \frac{e^{1.26}}{e^{1.26} + e^{0.82} + e^{0.43}} \\ \frac{e^{0.82}}{e^{1.26} + e^{0.82} + e^{0.43}} \\ \frac{e^{0.43}}{e^{1.26} + e^{0.82} + e^{0.43}} \end{bmatrix} = \begin{bmatrix} .48 \\ .31 \\ .21 \end{bmatrix}$$

with the probability that the input observation belongs to class 1, 2 or 3 being .48, .31 and .21, respectively.

### A.2.2.3 Optimisers

In the context of training neural networks optimisers are used to minimise the cost function. The gradient descent optimisation method follows the negative gradient of the cost function wrt the model's parameters to adjust the weights and biases. A limitation of the method is that it uses the same learning rate for every input variable.

The *AdaGrad* (Duchi et al., 2011) extension to gradient descent adds a self-adaptive learning rate for each parameter of the neural network model. It modifies the learning rate  $\eta$  at each step  $l$  for each parameter  $\theta_i$  based on past values of its gradient. Specifically, the modification to  $\eta$  is inversely proportional to the square root of the sum of gradients observed for the parameter since the beginning of training. AdaGrad removes the need to manually tune the learning rate but in practice it has been found that the AdaGrad learning rate converges to zero prematurely because of excessive accumulation of past gradients (Goodfellow et al., 2016).

RMSProp is based on AdaGrad and modifies the learning rates based on an exponentially decaying average of the past values of the gradients (Goodfellow et al., 2016). This modification has the effect of discarding historic values of gradients that are too old. RMSProp is an effective optimiser for neural networks in practice (Goodfellow et al., 2016). The ADAM optimiser (Kingma and Ba, 2014) adds momentum to RMSProp.

Adadelta (Zeiler, 2012) is also based on AdaGrad but instead of accumulating all historic gradients for a parameter it keeps a running average calculated from (a) the value of the parameter gradient at time  $t$  and (b) the running average computed at step  $t - 1$ .

### A.2.2.4 Convolutional Neural Network Architecture

Convolutional Neural Networks (Cun et al., 1990) or CNNs, are artificial neural networks that learn to

- (a) recognise and extract features from tensor inputs (multidimensional arrays) and
- (b) perform classification using the learned features jointly, in one unified neural network architecture.

CNNs are able to successfully capture and make use of spatial and temporal dependencies in

tensor inputs (such as color images<sup>6</sup> and text<sup>7</sup>) through the use of filters. The filters reduce the dimensionality of the input in a way that extracts useful features.

Convolutional Neural Networks use an operation called *convolution* instead of matrix multiplication used in MLPs in at least one of their layers (Goodfellow et al., 2016). Informally, a convolution operation weights a neighbourhood of spatially or temporally related inputs (e.g., adjacent pixels in an image or sensor readings over time) based on the neighbourhood location relative to its containing tensor. Using the weighted neighbourhood the convolution returns a convolved value for that neighbourhood's position.

More formally, the convolution operation *conv* is defined as:

$$\text{conv}(t) = \sum x(a)w(t - a) = (x * w)(t)$$

where  $x$  is the function that generates the inputs,  $w$  is a function that assigns the weight for inputs at distance  $a$  from time  $t$ . In the terminology of CNNs, the function  $w$  is referred to as the *kernel* and the output of convolution as the *feature map*. Convolution is typically denoted by the operator  $*$ .

Many kernels can be applied in parallel and the weight matrix  $w$  of each becomes part of the parameter set of the model (i.e., they are learned).

Let us now look at an example of convolution. The  $2 \times 2$  kernel in Figure A.7 is slid across the input one column at a time. The application of the kernel to the red, green and blue neighbourhoods of the input produces the first row of the output. The bottom row of the output is produced by applying the kernel across  $2 \times 2$  neighbourhoods formed around the last two rows of the input. The sliding size of the kernel across the input is called the *stride* and the stride for Figure A.7 is equal to 1.

A *convolutional layer* typically has three stages. In the first stage, one or more convolutions are performed and the activations that come out of this stage are the linear outputs of the convolutions (i.e., linear activations). It is during this stage that candidate features are extracted. In the second stage each activation from the first stage is passed through a non-linear activation function (ReLU from Section A.2.2.2 a typical choice). This stage learns to detect features automatically from the input. In the final stage, a *pooling function* (a form of kernel designed to summarise regions of the feature map) is applied to the activation map produced in the second stage and is specified as a hyperparameter to the model.

A pooling operation commonly used with CNNs is *maxpooling*. Maxpooling calculates the maximum value for each sliding neighbourhood of the feature map. As shown in Figure A.8, the output of maxpool for each neighbourhood becomes a corresponding component in the output.

---

<sup>6</sup>input tensor is a pixel grid with multiple color channels

<sup>7</sup>input tensor is a sequence of word embeddings representing a sentence

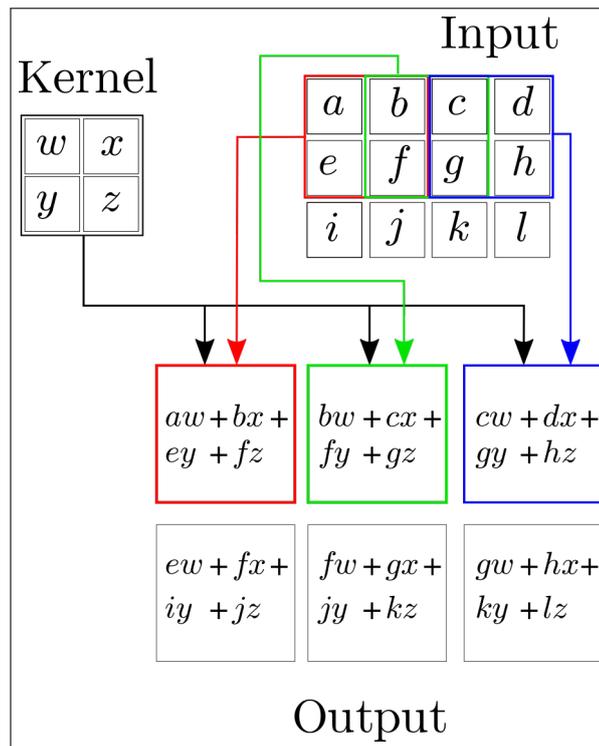


Figure A.7: Application of a  $2 \times 2$  kernel to 2-dimensional input (adapted from Goodfellow et al. (2016)).

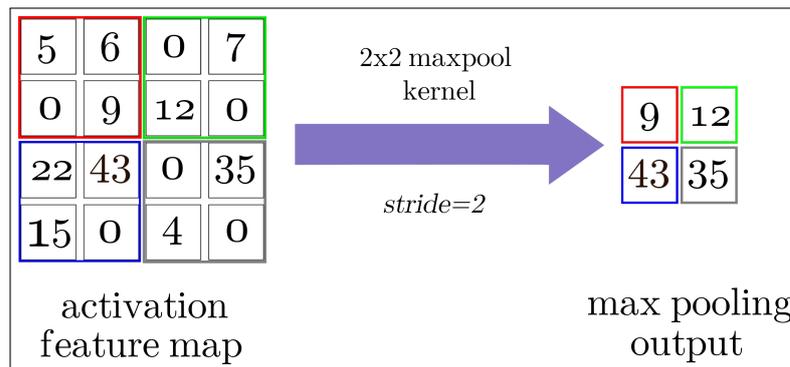


Figure A.8: Example of the application of a  $2 \times 2$  maxpooling kernel.

Pooling applied in the third stage reduces the dimensionality of the learned features. More importantly, pooling also makes the output representation of the features invariant to small translations in the input. This invariance gives CNNs the ability to recognise the presence of input features or properties without having to know where in the input the features occur (Goodfellow et al., 2016).

The output of a convolutional layer is a learned description of its input that can be fed-forward to other layers, such as fully-connected layers (MLP), to form classifiers as part of a unified convolutional neural network architecture. CNNs can be trained end-to-end using backpropagation.

### A.2.2.5 Long Short-term Memory (LSTM) Architecture

The Long Short-term Memory neural network or LSTM (Hochreiter and Schmidhuber, 1997b) is a deep neural network architecture particularly suited for sequences-to-sequence tasks, such as machine translation, and for predicting the next object in a sequence. For example, in natural language processing LSTMs are useful for building language models that predict word  $t$  in a sentence, given the words already generated at positions  $t - 1, \dots, 1$ .

LSTM is a special instance of the *recurrent neural network* (RNN) architecture. In the feedforward neural network, introduced at the beginning of this section, inputs flow through the hidden layers to the output in one direction. In RNNs information from the output layer can be looped back to become part of the input. I will motivate this looping mechanism using an example from natural language processing. Suppose we want to predict the next word in the sentence

“John saw \_”

from the word vocabulary { “John”, “Sue”, “called”, “spoke”, “to”, “saw” }. Good predictions for the next word in the sequence are “Alex” or “Sue”. However, a simple model that does not take into consideration the words already observed in the sequence might think that “John” and “saw” are also good candidates.

RNNs avoid bad predictions like the ones in the example by having a working memory of the last prediction (output predicting “John”) and using it in combination with the current input word (embedding vector for “saw”) to select the next word in the sequence.

In order to give an overview of RNNs and LSTMs, it is useful to take a high-level view and abstract the operation of connected layers into functional cells. Figure A.9 shows the functional cell of an RNN<sup>8</sup>. An RNN cell takes an input for time  $t$ ,  $X_t$  (e.g., an embedding vector for the word at position  $t$ ) and concatenates it to the output of the cell at time  $t - 1$ . The concatenated vector is then passed through a  $\tanh$  activation layer to produce the output for time  $t$ .

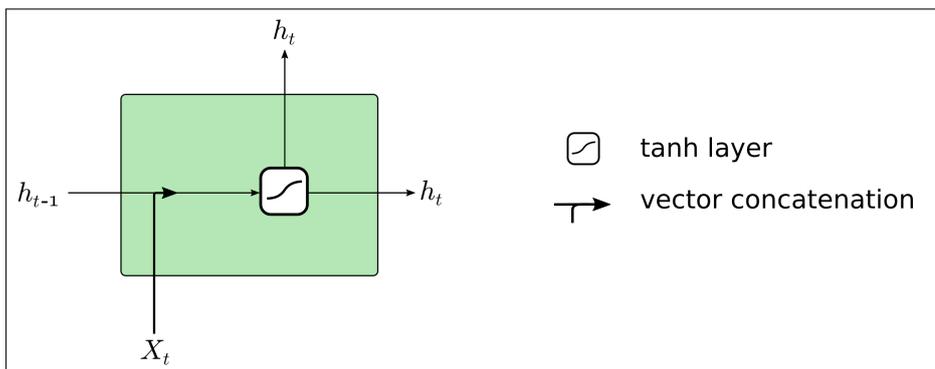


Figure A.9: Information flow in an RNN cell with a hyperbolic tangent activation layer.

<sup>8</sup>I omit the flow of biases in the diagram for simplicity.

In practice, RNN cells are unrolled into a connected chain, as shown in Figure A.10 and trained using backpropagation through time (Werbos, 1990).

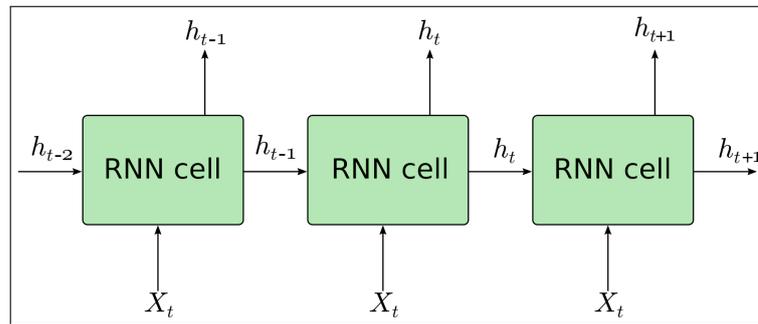


Figure A.10: RNN cells connected in a chain forming a deep neural network.

Despite the fact that RNNs can learn with arbitrarily long sequences, in practice it is difficult to train RNNs in tasks that require learning long-term temporal dependencies (Chung et al., 2014). Another issue is the *vanishing gradient problem*. RNNs use hyperbolic tangent as their activation and squeeze values in the  $-1$  to  $1$  range. The range of values of the gradient of  $\tanh$  are also small, between  $0$  and  $1$ . As gradients are calculated by the chain rule, small values for activations and errors are multiplied and further squeezed by  $\tanh$ . As a result, the gradients become almost zero and training of the model using backpropagation stops because subtracting near-zero gradients from the weights does not update the weights substantially.

The *exploding gradient problem* associated with RNNs is also of note. While following the chain rule (formula (ii) of backpropagation from Section A.2.2) multiplications with the transposed of the weight matrix are performed and if the values involved are large, the results become very large exploding the gradient.

LSTMs are kinds of RNNs with memory of long-distance dependencies. The LSTM cell, shown in Figure A.11, is composed of *gates* – functional units that regulate what information is remembered (or ignored) and utilised by the cell to make predictions. In Figure A.11 rectangular components are layers with sigmoid or  $\tanh$  activations, and circular components are operators. The operators product and add are element-wise operators on same-size vectors and the  $\tanh$  operator is a transformation performed on the individual components of a vector.

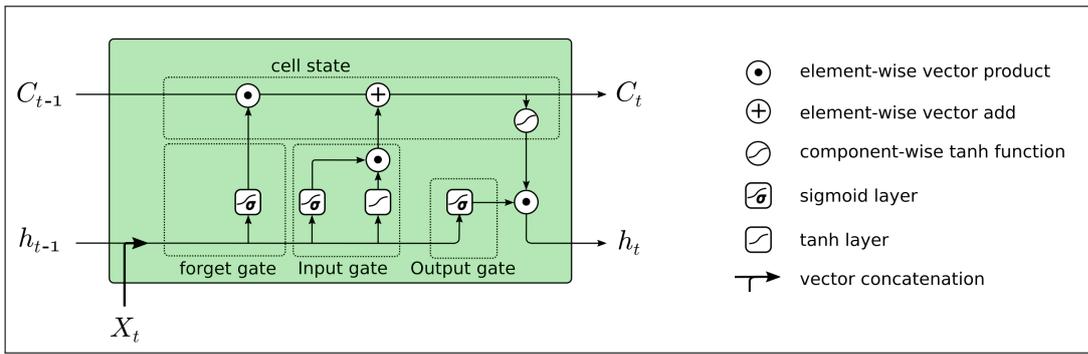


Figure A.11: The LSTM cell, its gates (denoted by dotted rectangles), components and information flow in the cell.

There are three inputs to the LSTM cell as shown in Figure A.11. The first is the hidden state,  $h_{t-1}$ , the output of the cell at time  $t - 1$ , which contains information about the input and output of the cell at time  $t - 1$ . The second input,  $C_{t-1}$ , is the state of the LSTM cell at time  $t - 1$  (the previous cell state). The third input,  $X_t$ , is the observation at time  $t$ .

Before I discuss each gate, I will first introduce the idea of gating. For this purpose, I will sometimes refer to the output from sigmoids as the gate and the output from tanh as the signal in this paragraph. The element-by-element multiplication operations in Figure A.11, with one input coming from a sigmoid and the other coming from tanh, act as gates. Intuitively, the gate input to  $\odot$  assigns importance to each component of the signal: when the gate is 0 the signal is blocked (output of  $\odot$  is 0) and for values smaller than 1, the signal is attenuated (output is smaller than the original value of the signal).

At time  $t$ , the cell starts by concatenating the inputs  $h_{t-1}$  and  $X_t$  and passes four copies of the result to three gates. The *forget gate* is the first unit in the LSTM cell. It passes information from the previous state and current observation through a sigmoid layer to decide what information from the previous state  $C_{t-1}$  is to be retained.

The *input gate* decides what information is going to be stored in the cell's state. The sigmoid layer decides what information should be updated in the state and the tanh layer creates the new values. Combining the outputs of the two layers with the  $\odot$  operation (as explained earlier) creates an intermediate cell state  $\bar{C}_t$ . The previous state  $C_{t-1}$  is updated with information in  $\bar{C}_t$  using the  $\oplus$  operation to produce the current state,  $C_t$ .

The *output gate* decides the output of the LSTM cell (current hidden state,  $h_t$ ). This decision is made by a sigmoid layer which gates information in the current state  $C_t$  (after the values of  $C_t$  are squashed into the  $-1$  to  $1$  range by a tanh transform) to filter out information that is no longer relevant at time  $t$ .

Sequence-to-sequence models can be constructed by unrolling LSTM cells and chaining them to form a sequence of the desired length, as shown in Figure A.12. Unrolled LSTMs can be trained using backpropagation through time (Werbos, 1990).

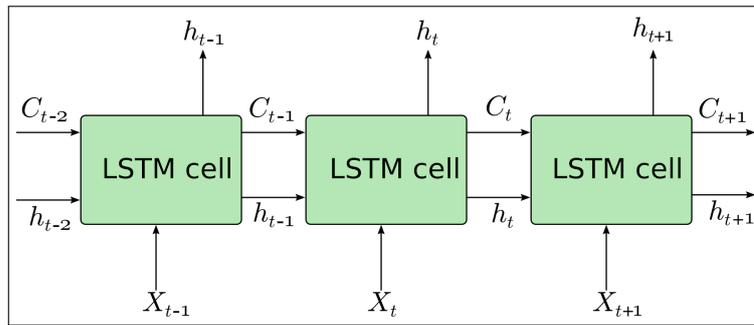


Figure A.12: Unrolled LSTM cells connected in a chain.

One extension to LSTMs that is relevant to this thesis is the bidirectional LSTM (BiLSTM). In the BiLSTM architecture, shown in Figure A.13, the input sequence is passed forward-to-backwards but also backwards-to-forwards.

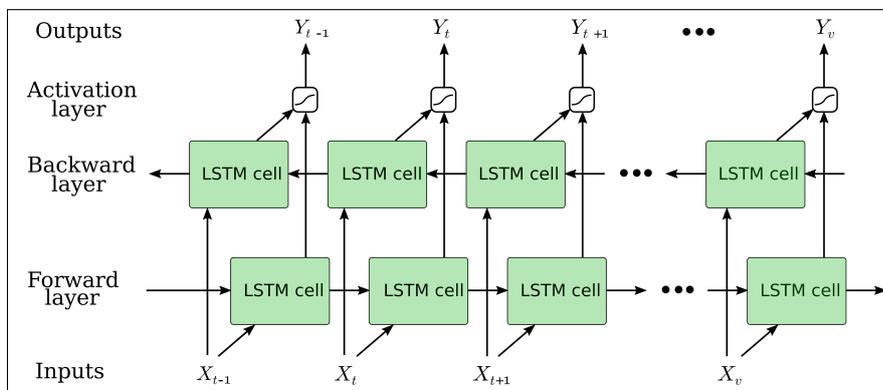


Figure A.13: Architecture of the bidirectional LSTM architecture.

The BiLSTM architecture allows the network to learn from inputs coming from past and future elements of the input sequence.

### A.3 Significance Testing

In the context of Information Retrieval, significance testing is used to determine whether the difference in performance (e.g., with respect to MAP) between the runs of two retrieval systems is statistically significant. The runs of the two systems being compared are considered to be *paired samples* – the score obtained by one system for a particular topic is paired to that of the other system for the same topic. Therefore, significance testing between two system runs requires a *paired difference test* designed to assess whether the population mean (in terms of performance) of the two runs is different.

Formally, a paired difference test is carried out in the form of a two-tailed paired hypothesis test with two disjoint hypotheses:

$H_0 : \mu_1 = \mu_2$  or equivalently  $\mu_1 - \mu_2 = 0$

there is no difference in statistic  $\mu$  between the two samples

$H_1 : \mu_1 \neq \mu_2$  or equivalently  $\mu_1 - \mu_2 \neq 0$

there is a difference in statistic  $\mu$  between the two samples

| Decision                              | Truth        |               |
|---------------------------------------|--------------|---------------|
|                                       | $H_0$ True   | $H_1$ True    |
| <b>Reject <math>H_0</math></b>        | Type I Error | Correct       |
| <b>Do Not Reject <math>H_0</math></b> | Correct      | Type II Error |

Table A.1: Outcomes of a hypothesis test.

Testing for significance reflects the likelihood that the difference observed in the two samples will transfer to the query population (as represented by the test collection). The null hypothesis ( $H_0$ ) is traditionally the simplest of the two hypotheses, while the alternative hypothesis ( $H_1$ ) is usually the composite hypothesis. In order to carry out a hypothesis test, an experimenter must fix two inversely related parameters –  $\alpha$  and  $\beta$ . Parameter  $\alpha$ , also referred to as the *significance level*, reflects the probability of a Type I error – a false positive in the test. That is, concluding that there exists significant difference between the populations when in fact there is none. As shown in Table A.1,  $\alpha$  is:

$$\alpha = P(\text{Type I Error}) = P(\text{Reject } H_0 | H_0 \text{ is True})$$

Concluding that the difference in performance between systems  $A$  and  $B$  is significant at confidence level  $100 \times (1 - \alpha)$  indicates that there is  $\alpha$  probability that the conclusion is a false positive. Conversely, if the significance test is reliable and for larger query samples, one system will be better than the other in 95% of comparisons. Typical values for  $\alpha$  are 0.05 (95% confidence) and 0.01 (99% confidence).

The parameter  $\beta$  is the probability of a false negative in the test – concluding that there is no significant difference between the populations when in fact a significant difference is present. From table A.1:

$$\beta = P(\text{Type II Error}) = P(\text{Do Not Reject } H_0 | H_1 \text{ is True})$$

The *power* of a test,  $(1 - \beta)$ , is the probability of detecting a significant difference given that a

difference is present:

$$1 - \beta = 1 - P(\text{Type II Error}) = P(\text{Reject } H_0 | H_1 \text{ is True})$$

A more powerful test is more likely to detect significant difference between two runs, if a difference exists, than a less powerful test. A less powerful test requires a larger sample than a more powerful test in order to achieve the same confidence level.

A plethora of *parametric* and *nonparametric* paired difference tests are available to the IR researcher. Parametric tests require that certain assumptions about the population from which the samples are drawn be satisfied. For example, one of the assumptions of the paired t-test is that the differences in the paired sample means are normally distributed. However, in IR it is seldom the case that scores (and their differences) follow a well-known statistical distribution. Nonparametric tests have more general requirements that are not related to the specific form of the distribution from which the samples are drawn (Siegel and Castellan, 1988).

For this purpose, IR researchers often rely on non-parametric tests for significance testing of paired runs between two systems on the same test collection. The *Sign test* (Siegel and Castellan, 1988), *Wilcoxon signed ranks test* (Wilcoxon, 1945; Siegel and Castellan, 1988) and *Permutation test* (Pitman, 1937a; Pitman, 1937b; Pitman, 1937c; Scheffe, 1943; Fisher, 1970; Siegel and Castellan, 1988), also known as the randomisation test, are paired, nonparametric tests that can be used to test for significance in IR experiments.

The sign test has more general assumptions but is less powerful than the Wilcoxon and Permutation tests, the latter being the most powerful (Siegel and Castellan, 1988). Smucker et al. (2007) have shown that the Wilcoxon test is unreliable for tests involving MAP and is more likely to produce Type II errors. In contrast, the authors have found the Permutation test to be reliable with MAP and its derivatives and recommend its use for significance testing in IR evaluations involving MAP. Therefore, for my experiments I will be using the paired Permutation test to measure significance, which I will describe in the paragraphs that follow.

### **Permutation (or Randomization) Test**

Given paired runs from two systems,  $A$  and  $B$  of length  $N$ , and a function  $\mu$  implementing measure  $M$  on the entirety of each run, the Permutation test begins by assuming that the null hypothesis is true. That is, it proceeds as if both runs come from the same equally-performing system (difference in  $M$  between the runs is 0). Under these assumptions, the Permutation test evaluates the probability that the observed difference in  $M$  between the runs has been obtained by random chance.

This is done by re-sampling the observed data repeatedly as follows. For each paired observation in the original runs,  $a_i$  and  $b_i$ , a coin is flipped. If the outcome is 1, then the score for  $b_i$  is moved to  $a_i$  and vice-versa. Otherwise,  $a_i$  and  $b_i$  remain unchanged. This process is repeated  $R$  times, with  $R$  a parameter of the Permutation test, yielding  $R$  permuted paired samples of

|                              | System A | System B | Coin Toss | Permuted A | Permuted B |
|------------------------------|----------|----------|-----------|------------|------------|
| Topic 1                      | 0.01     | 0.1      | 1         | 0.1        | 0.01       |
| Topic 2                      | 0.03     | 0.15     | 0         | 0.03       | 0.15       |
| Topic 3                      | 0.05     | 0.2      | 0         | 0.05       | 0.2        |
| Topic 4                      | 0.01     | 0.08     | 1         | 0.08       | 0.01       |
| Topic 5                      | 0.04     | 0.3      | 0         | 0.04       | 0.3        |
| Topic 6                      | 0.02     | 0.4      | 1         | 0.4        | 0.02       |
| Observed MAP                 | 0.0267   | 0.205    |           | 0.117      | 0.105      |
| Absolute Observed Difference | 0.178    |          |           | 0.0017     |            |

Table A.2: Example runs from two systems and an example iteration of the re-sampling process (last two columns).

the original runs. For each generated pair of permuted runs, the statistic  $\mu$  is computed for each system and the difference between them is obtained. This process models the assumption that if the two runs are indeed the same, then the paired re-assignments should have no impact on the difference in  $M$  between the samples.

Once re-sampling is complete, the probability of observing the difference between the original runs by chance can be approximated by:

$$p = \frac{s + 1}{R + 1} \quad (11)$$

where  $s$  is the number of permuted samples with difference in  $M$  higher than the one observed in the original runs. In formal statistical terms, the value of  $p$  is known as the  $p$ -value of the Permutation test. It is the probability of finding the observed (or more extreme) results under the null hypothesis,  $H_0$ . The null hypothesis is rejected if  $p < \alpha$  and the test concludes that there is evidence in support of the difference between the runs being significant (the alternative hypothesis,  $H_1$ , must be true).

For large  $N$  it may be infeasible to enumerate all possible permutations of the observations. In such cases, the Permutation test can be applied by sampling a sufficiently large number of permuted replicates. This instance, referred to as the *approximate Permutation test* or *Monte Carlo Permutation test*, produces an estimate  $\hat{p}$  of the true  $p$ -value. Methods for efficiently determining the number of re-sampled permutations necessary to guarantee that the approximate  $p$ -value is correct have been developed (Gandy, 2009).

**Example.** Consider the paired runs presented in Table A.2. There are  $2^6$  possible permutations for the pairs, one of which is represented by the last three columns of Table A.2. An exhaustive re-sampling of the runs reveals that under the null hypothesis, 2 out of 64 permutations are equal or larger than the observed difference in MAP, 0.178. By applying formula 11 with  $s = 2$  and  $R = 64$  we obtain  $p$ -value = 0.0462. Since this is smaller than  $\alpha = 0.05$ , we reject the

null hypothesis at the stated confidence level and conclude that there is significant difference in retrieval efficiency between systems  $A$  and  $B$ .

In my experiments, presented in Chapter 7, I have used the implementation of the approximate Permutation test from the `EnvStats` R package with 5000 permuted replicates to carry out tests for significance between model runs.



# Appendix B

## Test Collection Analysis Examples

### B.1 Example of the Application of my Process

In this section I provide examples of the application of my process to MathOverflow threads. Although the cases I will discuss here represent four unforeseen challenges, my process has been flexible enough to address them without the need for domain expertise.

The first challenge comes from threads about *open problems*, where the questioner has no knowledge of the fact that his/her information need relates to an open mathematical problem. Questions that involve open (sub-)problems can be included in my test collection provided that the relevant citations are, according to experts on the site, the best available resources on the open problem.<sup>1</sup>

The second challenge is exemplified in one case, thread 61615, where the information seeker gave confirmation of relevance to multiple candidate answers in the PA comments:

*I've seen four answers (by Gjergji, Mark, Steve, and Nishant), all very helpful - thank you very much. I'm accepting Gjergji's on the grounds that it was posted first.*

In response to this comment, I checked all answers for citations involving MREC documents. Only the accepted answer included such citations.

The third challenge concerns ambiguous confirmation of relevance by the questioner in the PA, as exemplified in thread 74843. The questioner for the thread commented “thank you, that’s exactly what I need” in the PA comments. However, it was unclear whether the comment was for (a) the MREC resource cited in the answer when it was originally authored, or (b) one of the non-MREC citations added later via an edit. The ambiguity could be resolved in this case by examining the timestamps of posts: the answer, originally posted on 11/9/2011 6:26, was edited on 11/9/2011 at 18:03. Since the questioner’s confirmation was posted on 11/9/2011 at 7:57, I concluded that the acknowledgement must have been for the cited MREC document.

The fourth challenge affects my ability to evaluate typed retrieval. There are two questions where formulae and variables were not typeset in  $\text{\LaTeX}$ , but left in the body as plain text (ASCII).

---

<sup>1</sup>This happened to be the case in three threads of my collection (40887, 120893, 142938).

As a result, these formulae and variables could not be automatically converted to Presentation MathML, or SLTs. This introduces a practical consideration because the variables and types of some query expressions cannot be discovered. Arguably, I should have introduced  $\text{\LaTeX}$  for formulae not properly typeset and/or excluded these questions. At the time I constructed my test collection, however, my goal was to create a data set that mirrored the real world as closely as possible, so I decided to keep them as is.

Information collected in every step of my method was recorded on two forms: one used to note information regarding a MO question (Figure B.1 shows an example) and another to collect information about its accepted answer (Figure B.2 shows an example). The full-scale example is attached in Appendix B.2.

**Data collection form for an MO question**

| SEQUENCE ID | THREAD ID | TAGS   MREK citations                 | 1st PASS | ATTACHED | USEFULNESS     |
|-------------|-----------|---------------------------------------|----------|----------|----------------|
| 33.5        | 45702     | ct.category-theory<br>homotopy-theory | ✓        |          | ✓<br>posterboy |

| Sub-questions  |   | Query | Prelude | Key Sentences  |                                |                                   |  |   |   |   |   |   |  |  |                                 |  |                                  |  |  |  |   |   |
|--|---|-------|---------|--|--------------------------------|-----------------------------------|--|---|---|---|---|---|--|--|---------------------------------|--|----------------------------------|--|--|--|---|---|
| ①  | ②   | 2     | 10      | <p>How tall is a tree? ... of their classes of ...</p> <p>I'm looking for the name of a certain n-categorical definition:</p> <p>"(Someone explained it to me a couple of years ago. I remember the definition but not the name. Without the name it's difficult to search for a citation. I want the citation in order to explain something without needing a paper.)"</p> <p>"For background, consider the Moore (any space) of loops of length <math>r</math> (that is, parameterized by the interval <math>[0, r]</math>). We have a strictly associative composition <math>\circ</math> on <math>\Omega_r X</math>."</p> <p>"The main idea of an 'n-categorical' n-category is to imitate these ideas in higher dimensions. The n-categorical one is parameterized by <math>n</math>-dimensional rectangles with sides of lengths <math>l_1, \dots, l_n</math>."</p> <p>"Gluing rectangles together gives a different strictly associative ways to compose n-morphisms."</p> <p>"Question: What is 'n-cats' about?"</p> <p>"Bonus question: what's the best (or only) citation for this idea?"</p> <p>"EDIT: It turns out the definition I was trying to remember is unpublished work of Ulrich Tillmann. But the name Bruce Breen linked to in David Roberts' answer is pretty similar (for my purposes, at least)."</p> |                                |                                   |  |   |   |   |   |   |  |  |                                 |  |                                  |  |  |  |   |   |
| <p><b>CHARACTERISTICS</b></p> <table style="width: 100%; border: none;"> <tr> <td><input type="checkbox"/> Empty</td> <td><input type="checkbox"/> Interest</td> </tr> <tr> <td><input type="checkbox"/> Proof Request</td> <td><input type="checkbox"/> Object satisfying properties</td> </tr> <tr> <td><input type="checkbox"/> Compute Object</td> <td><input type="checkbox"/> Construct Object</td> </tr> <tr> <td><input type="checkbox"/> Universal Object</td> <td><input type="checkbox"/> Universal property of object class</td> </tr> <tr> <td><input type="checkbox"/> What is the name...</td> <td></td> </tr> <tr> <td><input type="checkbox"/> How...</td> <td></td> </tr> <tr> <td><input type="checkbox"/> When...</td> <td></td> </tr> <tr> <td><input checked="" type="checkbox"/> Name problem</td> <td><input type="checkbox"/> Counted Example Request</td> </tr> <tr> <td><input checked="" type="checkbox"/> Reference/Quotation</td> <td><input type="checkbox"/> Comparison request</td> </tr> </table> <p><b>Properties</b>      <b>Objects</b></p> <p>n-category      Moore, Loop space</p> <p>loops      interval</p> <p>strictly associative composition      Noticeable/Repeating Patterns</p> <p>n-morphisms      ① Love this question. "I have the defn but I can't search for resources because I need a name" → <math>\cup</math> indicates my purpose!</p> <p>n-dimensional rectangles      ② Infoscience confirms usefulness of citation in Question Edit.</p> <p><b>Cluster Labels</b></p> |   |       |         |  | <input type="checkbox"/> Empty | <input type="checkbox"/> Interest | <input type="checkbox"/> Proof Request | <input type="checkbox"/> Object satisfying properties | <input type="checkbox"/> Compute Object | <input type="checkbox"/> Construct Object | <input type="checkbox"/> Universal Object | <input type="checkbox"/> Universal property of object class | <input type="checkbox"/> What is the name... |  | <input type="checkbox"/> How... |  | <input type="checkbox"/> When... |  | <input checked="" type="checkbox"/> Name problem | <input type="checkbox"/> Counted Example Request | <input checked="" type="checkbox"/> Reference/Quotation | <input type="checkbox"/> Comparison request |
| <input type="checkbox"/> Empty   | <input type="checkbox"/> Interest                           |       |         |  |                                |                                   |  |   |   |   |   |   |  |  |                                 |  |                                  |  |  |  |   |   |
| <input type="checkbox"/> Proof Request   | <input type="checkbox"/> Object satisfying properties       |       |         |  |                                |                                   |  |   |   |   |   |   |  |  |                                 |  |                                  |  |  |  |   |   |
| <input type="checkbox"/> Compute Object  | <input type="checkbox"/> Construct Object                   |       |         |  |                                |                                   |  |   |   |   |   |   |  |  |                                 |  |                                  |  |  |  |   |   |
| <input type="checkbox"/> Universal Object  | <input type="checkbox"/> Universal property of object class |       |         |  |                                |                                   |  |   |   |   |   |   |  |  |                                 |  |                                  |  |  |  |   |   |
| <input type="checkbox"/> What is the name...   |   |       |         |  |                                |                                   |  |   |   |   |   |   |  |  |                                 |  |                                  |  |  |  |   |   |
| <input type="checkbox"/> How...  |   |       |         |  |                                |                                   |  |   |   |   |   |   |  |  |                                 |  |                                  |  |  |  |   |   |
| <input type="checkbox"/> When...   |   |       |         |  |                                |                                   |  |   |   |   |   |   |  |  |                                 |  |                                  |  |  |  |   |   |
| <input checked="" type="checkbox"/> Name problem   | <input type="checkbox"/> Counted Example Request            |       |         |  |                                |                                   |  |   |   |   |   |   |  |  |                                 |  |                                  |  |  |  |   |   |
| <input checked="" type="checkbox"/> Reference/Quotation  | <input type="checkbox"/> Comparison request                 |       |         |  |                                |                                   |  |   |   |   |   |   |  |  |                                 |  |                                  |  |  |  |   |   |
| <p><b>COMMENTS</b></p> <p>① This is a 'posterboy' question!</p>  |   |       |         |  |                                |                                   |  |   |   |   |   |   |  |  |                                 |  |                                  |  |  |  |   |   |

Figure B.1: Data collection form for the question part of a MathOverflow thread.

## Data collection form for an MO answer

The form is titled "The Answer" and contains the following sections:

- Citations:** A table with columns "Sub-answers", "Total cite", and "MREC". The "Total cite" column contains the number "1". Below this is a table for "Cited MREC Docs" with columns "A" and "ID". The "ID" column contains "0909.2212".
- Key Sentences:** A text area containing a handwritten quote: "Ravi B. Brown has a related idea contained in this article 'More hyperrectangles on a space form a strict cubical category' [arXiv:1108.0487]. If you are instead thinking of a globular n-category, the closest I know of is a Trimble n-category, but that doesn't use Moore paths, but paths of length 3 and the Assoc-co-category structure on  $[0, 1]$ ."
- Characteristics:** A section with checkboxes for "Direct" (checked), "Indirect", "Use of Analogues", "General", "Partial", "Spec.", and "Solution Involves Named Objects/Concepts". There are also fields for "RELEVANCE" (1+1) and "DISTANCE" (2).
- How to get Answer and/or Role of citation to obtaining answer:** A text area containing the handwritten note: "Read the ref to get answer..."
- Properties and Objects:** A table with columns "Properties" and "Objects". The "Objects" column contains handwritten terms: "More hyperrectangles", "strict cubical category", "Trimble n-category", "Moore paths", and "Assoc-co-category structure".
- Role of MREC citations:** A section with checkboxes for "Part of a direct answer" (checked), "Background material", "More interesting", "Unclear/Too General", and "Starting Point".
- COMMENTS:** A text area containing handwritten text: "Cited [MREC-1] contains a structure similar to that requested as confirmed by infoseeker in Q edit and PA comments: 'Thanks, that's helpful. The paper by Brown matches what I remember pretty well, but I thought there was some other name for this.' ACCEPT!"

Annotations on the right side of the form:

- A dashed box around the citations table is labeled: "Recording citations: Total and number of citations in MREC. Individual arXiv IDs recorded."
- A dashed box around the "How to get Answer..." section is labeled: "Detailed proof/answer derivation trace (sometimes on separate sheet)."
- A dashed box around the "COMMENTS" section is labeled: "Answer comments used to justify final decision and note interesting patterns in the answer (such as PA confirmation)."

Figure B.2: Data collection form for the answer part of a MathOverflow thread.

## **B.2 Full-scale Example Forms**

| SEQUENCE ID | THREAD ID | TAGS   MREC citations  | 1st PASS | ATTACHED | USEFULNESS |
|-------------|-----------|--|----------|----------|------------|
| 56          | 133673    | qa.quantum-algebra<br>mp.mathematical-physics<br>quantum-groups<br>quantum-field-theory<br>quantum-topology. | ✓        | 2        |            |

## The Question

| Sub-questions  | Query  | Prelude | Key Sentences  |
|--|--|---------|--|
| ① ②  | 2  | 7       | <p>"How can one see that the Turaev-Viro model as a discretization of the path integral?"</p> <p>"How does the connection between these two state sums go?"</p> <p>"Turaev-Viro defined an invariant of three-manifolds <math>M</math> denoted <math>TV(M)</math>, which was subsequently shown by ... to coincide with <math> WRT(M) ^2</math> where <math>WRT(M)</math> is the Witten-Reshetikhin-Turaev invariant of <math>M</math>."</p> <p>"Both invariants depend on a choice of quantum group <math>U_q(\mathfrak{g})</math> and a root of unity <math>q</math>."</p> <p>"Witten's 'physics definition' of <math>WRT(M)</math> ... is via an 'integral over all <math>\mathfrak{g}</math>-connections on <math>M</math>'. I have read in many places that the Turaev-Viro model is essentially a discretization of the path integral."</p> <p>"(which only converges if we use a quantum group; the nonconvergent analogue for a classical group is the 'Ponzano-Regge model'.")</p> <p>"The definition of <math>TV(M)</math> is a 'state sum' over all assignment of representations of <math>U(\mathfrak{g})</math> to each of the edges of (a fixed triangulation of) <math>M</math>."</p> <p>"On the other hand, one would naively expect that a discretization of the path integral would be a 'state sum' over all assignments of 'elements' of <math>U_q(\mathfrak{g})</math> to each of the edges of <math>M</math>."</p> |
| Characteristics  |  |         | Noticeable/Repeating Patterns  |
| <input type="checkbox"/> Binary <input type="checkbox"/> Existential<br><input type="checkbox"/> Proof Request <input type="checkbox"/> Object satisfying properties<br><input type="checkbox"/> Compute Object satisfying prop. <input type="checkbox"/> Construct Object satisfying properties<br><input checked="" type="checkbox"/> Universal corresp. between objects <input type="checkbox"/> Universal property of object class<br><input type="checkbox"/> What...<br><input checked="" type="checkbox"/> How... can one see that A is B?<br><input type="checkbox"/> When...<br><input type="checkbox"/> Name Problem <input type="checkbox"/> (counter) Example Request<br><input type="checkbox"/> Reference/Overview <input type="checkbox"/> Comparison request |  |         |  |
| Properties   | Objects  |         |  |
|  | Witten-Reshetikhin-Turaev invariant,<br>quantum group<br>root of unity.<br>$\mathfrak{g}$ -connection,<br>path integral<br>state sum |         |  |
| Cluster Labels   |  |         |  |
|  |  |         |  |

### COMMENTS

① Essentially, the answer to ② should be part of the "big" answer for ①.

# The Answer

|             |            |      |
|-------------|------------|------|
| Sub-answers | Total cite | MREC |
| ① ②         | 1          | 1    |

Key Sentences

"An explanation exactly as I was looking for appears at the beginning of section III of this paper [MREC]."

Cited MREC Docs

|   |                |
|---|----------------|
| A | hep-th/0401076 |
|---|----------------|

Characteristics

Direct       Use of Analogues

Indirect

|                    |       |   |
|--------------------|-------|---|
| RELEVANCE DISTANCE | ① + 1 | 1 |
|--------------------|-------|---|

General

Partial

Spec.

Solution involves Named Objects/Concepts

Answer includes Examples/Counterexamples

| Properties | Objects                |
|------------|------------------------|
|            | connection field       |
|            | continuum partition    |
|            | Pontryagin-Kegge model |
|            | triangulation          |
|            | frame field            |
|            | Lie algebra class.     |
|            | edges (of a triang.)   |
|            | discretized partition  |
|            | dual edges.            |

Role of MREC citations

Part of is direct answer       Unclear/too General

Background material       Starting Point

More information       Other

How to get Answer and/or Role of citation to obtaining answer

- ① The answer DOES NOT CONTAIN some tech-terms that were present in the question:  
g-connections, quantum-group, invariants, representations, path integral, root of unity
- ② The answer contains the following technical terms that have been present in the question:  
state sum, discretization, edges, triangulation (of a manifold),
- ③ Observations ① and ② are not surprising given some thought on how these keyword fit into the picture.
- ④ See Attachment (2) for a reformulation of question and answer.

COMMENTS

① Sub-answer ② follows from subanswer ①

② Infoserener is the one answering the question. Accept because Infoserener ~~has~~ cites [MREC-1] section III as relevant.

| SEQUENCE ID | THREAD ID | DESCRIPTION  | ATTACHMENT | PAGE X OF Y |
|-------------|-----------|--|------------|-------------|
| 56          | 133673    | Clarification of question and answer in step-by-step manner. | 2          | 1           |

### 1. The Question

- $TV(M)$  is an invariant of three-manifolds  $M$ .
- $TV(M) \equiv |WRT(M)|^2$  according to Kevin Walker.
- $WRT(M)$  is the Witten-Reshetikhin-Turaev invariant of  $M$ .
- Both  $TV(M)$  and  $WRT(M)$  depend on a choice of quantum group  $U_q(\mathfrak{g})$  and a root of unity  $q$ .
- $WRT(M)$  is defined via "an integral over all  $\mathfrak{g}$ -connections on  $M$ "
- $TV(M)$  is essentially a discretization of the path integral

Converges if we use a quantum group

non-convergent analogue is the "Ponzano-Regge" model for a classical group.

≡

- A "state sum" over all assignments of representation of  $U_q(\mathfrak{g})$  to each of the edges of a fixed triangulation of  $M$ .
- Infoseeker expects that a discretization of the path integral would be a "state sum" over all assignments of "elements" of  $U_q(\mathfrak{g})$  to each of the edges of  $M$ .  
(a discretized connection of  $M$ )

## 2. The Answer

$M$ : a manifold

$\Delta$ : a triangulation of  $M$

$\Delta^*$ : the dual to  $\Delta$ .

$g_{e^*}$ : group elements

$f^*$ : dual faces.

Ponzano-Regge model is obtained from the continuum partition function

$$Z_M = \int D_\omega D_e \exp \left[ \frac{i}{16\pi G} \int_M \text{tr} (e \wedge F(\omega)) \right]$$

by

- ① Considering a triangulation  $\Delta$  (and its dual  $\Delta^*$ ) of  $M$ , and replacing the set of configuration variables by discrete analogs (in the spirit of lattice gauge theory).
- ② The connection field is replaced by group elements  $g_{e^*}$  associated to the edges  $e^*$  of  $\Delta^*$  AND representing the holonomy of the connection field along these edges.
- ③ The frame field is replaced by Lie Algebra elements  $X_e$  associated to the edges  $e$  of  $\Delta$  and representing the integration of  $e$  along these edges.
- ④ The curvature 2-form is now represented as group elements  $G_e$  living on the edges  $e$  (or dual faces  $f^*$ ) AND obtained as the ordered product of the group elements  $g_{e^*}$  for dual edges  $e^* \subset f^*$ , upon the choice of a starting dual vertex on the dual face.
- ⑤ The discretized partition function becomes

$$Z_{PR}(\Delta) = \left( \prod_{e^*} \int_{SU(2)} dg_{e^*} \right) \left( \prod_e \int_{SU(2)} dX_e \right) \exp \left[ i \sum_e \text{tr} (X_e G_e) \right]$$

- ⑥ One can then integrate over the  $X_e$  variables

$$Z_{PR}(\Delta) = \left( \prod_{e^*} \int_{SU(2)} dg_{e^*} \right) \left( \prod_e \delta(G_e) \right)$$

where  $\delta(g)$  is the delta function over the group.

| SEQUENCE ID | THREAD ID | DESCRIPTION | ATTACHMENT | PAGE X OF Y |
|-------------|-----------|-------------|------------|-------------|
| 56          | 133673    |             | 2          | 2           |

- ⑦ The Panzano-Regge partition function is recovered from ⑥ by expanding the  $\delta$  function using the Plancherel decomposition, and then integrating over  $\mathcal{G}_e^*$  using recoupling identities for  $SU(2)$ .
- ⑧ The result can be expressed as a summation over coloring of a product of  $G_j$  symbols:

$$Z_{PR}(\Delta) = \sum_{\{j_e\}} \prod_e d_{j_e} \prod_j \left\{ \begin{array}{ccc} j_{e_{t_1}} & j_{e_{t_2}} & j_{e_{t_3}} \\ j_{e_{t_4}} & j_{e_{t_5}} & j_{e_{t_6}} \end{array} \right\}$$

where

- the summation is over all edges of  $\Delta$  and the product of  $G_j$  symbols is over all tetrahedra.
- $d_j = 2j+1$  denotes the dimension of the spin  $j$  representation
- $e_t$  denotes the six edges belonging to the tetrahedra  $t$ .
- $j_e$  plays the role of a discrete  $\chi_e$ .

- ⑨ It is therefore interpreted as a length and the Panzano-Regge sum is a sum over all geometries supported by a triangulation!



## **Appendix C**

### **Instructions to Annotators for Gold-Standard Type List**

Thank you very much for agreeing to help!

## 1. INTRODUCTION

Linguistically, a mathematical type is (typically) a *noun phrase*, i.e. a phrase headed by a noun with possible pre-modification (e.g., adjective phrases), and post-modification (e.g., prepositional phrases). Types can also sometimes include proper nouns, for instance, those of an inventor of some artefact. From now on, strings in bold represent examples of Types.

Examples include:

- **Algebraically closed field**
- **Set of Eigen Values**
- **Ring of Polynomial Functions**
- **Klein Bottle**

Conceptually, types are phrases used to refer to mathematical *concepts*, *structures* and *objects* that can be instantiated in the discourse.

The term *structures* refers to algebraic structures, which are typically uniquely constructed by grouping other types together – i.e. tuples of other types. Typical examples include

- A **monoid** is a **set** together with a **binary operator** (i.e., a **function**): (Set, binop: (Set X Set) -> Set)
- A **Category** is a **tuple** composed of a **Set of Objects** and a **Set of Morphisms** between these objects (plus some properties).
- A **Vector space** composed of two sets and two binary operators.

Structures are usually instantiated in the discourse for the purpose of setting up the mathematical context or for mathematical argumentation. For example:

“A **Group** is a tuple composed of a **Set of elements** equipped with a **binary operator** over the elements of the set that satisfies closure, associativity, identity and invertibility. A **Group**  $G$  is said to be an **Abelian Group** iff the **binary operator** satisfies the **axiom of commutativity**.”

Named theorems, axioms, conjectures and corollaries are also types, provided they are not excerpts from a discourse block (e.g., Theorem 4.1.1). Examples include **axiom of commutativity**, **Tanyama-Shimura Conjecture**.

Any instantiable *object* is also a type. This typically occurs in sentences of the form

“Let <variable> be <type>” such as “Let  $x$  be a **Vector**”

where the variable “ $x$ ” can take part in argumentation in the discourse or in other mathematical expressions. On one hand, the term “Object” may refer to fundamental types (such as **Sets** and instances of sets such as the **Set of Real Numbers**, sequence), special geometric objects and shapes (polyhedron, triangle) and **Functions** (also referred to as **Maps** or **Mappings**).

Special instances of types are also types. In some cases, it is straight-forward to assert whether a type is a special instance of a super-type since the phrase that names a sub-type also includes the super-type.

For example,

A **Riemannian Manifold** is a special kind of **Manifold**

A **free unitary quantum group** is a special kind of (quantum) **Group**

In other cases, the super/sub-type relation is not so obvious:

a **Tetrahydon** is a subtype of **Shape**

a **Klein Bottle** is a kind of **Surface**

On the other hand, Objects can be abstract *mathematical concepts* that are instantiated in the discourse in the form of a more concrete type, such as an algebraic or fundamental type. For example, consider the following sentence:

“The **envelope of elliptic trajectories** is indeed an **Ellipse**”

Here, the **envelope of elliptic trajectories** is an instantiable concept (a higher-order type in the sentence) which happens to take the form of an **ellipse** (a concrete type)

More commonly, this occurs with concepts that are described by a number. e.g,

the **Cardinality of a set** is a **Number**.

Some mathematical concepts are not instantiable into variables or do not reduce to other concrete types. Examples include

1. Names of mathematical frameworks such as “Chaos”
2. Properties between objects defined by complex conditions (e.g., “multicollinearity”, “reducability”) 3. phenomena such as “oversampling”, “overfitting”
4. Abstract or informal ideas such as “satisfiability” and “undecidability”.
5. Procedures that are part of the act of doing mathematics (e.g., “proof by contradiction”)
6. Mathematical processes. For example, “differentiation” is not a type, but a “differentiator” is a type.

## 2. THE TASK

You have been given is a list of X phrases and your task is to determine whether or not each phrase is a *mathematical type* or not. I am asking you to decide whether the phrase as a whole is a type or not.

### General Guidelines

1. Please use the rules below and your knowledge of Mathematics to annotate each phrase on the sheet as *type* or *not a type*.
2. You may consult Wikipedia, encyclopedia of Math or any other mathematical resource while applying these rules. Try searching for these phrases by putting them in quotes if you can't find any hits.
3. Keep in mind that some things might have multiple definitions
4. When examining the phrases, please ignore the case and artefacts such as typos, missing spaces or apostrophes.
5. Whether a phrase is in singular or plural does not matter.
6. Please circle the answer you think is correct on the right-most column of the questionnaire.
7. Please try to provide a decision for all phrases.

## Rules

1. Any phrase or word referring to an algebraic structure is a type.
2. Any phrase or word that refers to commonly used objects in Mathematics but that are not formally defined. is a (fundamental) type. Examples include **Number**, **Set**, **Function** and **Matrix**.
3. Any subtype of a type is also a type:
  - (a) A phrase that modifies or otherwise specialises (e.g., using a proper noun) a type is also a type.  
Examples: **Riemannian Manifold**, **Abelian Group** and **free unitary quantum group**
  - (b) Any phrase that, to your knowledge, refers to an instantiable construct that has a known supertype is also a type. This applies to subtypes whose super-type is not obvious from the phrase:

**Klein Bottle** is a subtype of **Surface**

- (c) Specialised, named instances of types that do not explicitly state the type they specialise are types. Examples:  
“A **Jacobian**” refers to the **Jacobian Matrix**.
4. Any phrase that establishes a relationship between types (e.g., elementhood, membership and ownership) is also a type. This usually involves prepositional phrases. Examples:

**Ideal of a Ring**, **Point on the Plane**

5. Any type that is parameterised/parameterisable is also a type. Examples of parameterised types include

**2-Group**, **L-Function**, **G-Function**, **n-Curve**

Due to text tokenization artefacts, the strings you might encounter may be parameterised types that are missing their parameter. For example, you might see

**-Group**, **-Function**, **-Curve**

Please ignore these artefacts and mark these strings as types as well.

6. Named theorems, axioms, conjectures and corollaries are types, provided they are not headers of sections such as “Theorem 4.1.1”. Examples include

**Axiom of commutativity**, **Tanyama-Shimura Conjecture**

7. Abbreviations or phrases with abbreviations that refer to types are also types. For example,

**KL-Divergence**, **KLD**, **kl divergence**

8. Mathematical concepts that, to your knowledge, are instantiable in the discourse and can take the form of concrete types, such as algebraic structures or fundamental types, are types.

**Cardinality of a set** is a **Number**.

**Envelope of elliptic trajectories** is indeed an **Ellipse**.

9. Names of mathematical frameworks such as “Chaos” are not types.
10. Proper nouns that are names of authors, software packages or scientific/mathematical disciplines are not types.
11. Properties between objects defined by complex conditions (e.g., “multicollinearity”, “reducibility”) are not types.
12. Phenomena such as “oversampling” and “overfitting” are not types.
13. Abstract, uninstantiable or informal ideas such as “satisfiability” and “undecidability” are not types.
14. Procedures that are part of the act of doing mathematics (e.g., “proof by contradiction”) are not types.
15. Mathematical processes, such as “differentiation”, are not types.

# Appendix D

## Supplementary Material and Results for Chapter 7

### D.1 Section 7.1

#### D.1.1 Comparative Results of TypeExp model

In chapter 7, I report the results of the TypesExp model in Figure 7.3 (page 168, repeated here as Figure D.2 for convenience). These are the results I achieved with the full model with the full type dictionary of 1.23 million types. The results are stronger than those published in (Stathopoulos and Teufel, 2016), because at the time, I did not have access to the full dictionary yet and instead operated with the seed dictionary of 10,601 types only.

For completeness, I am repeating the 2016 results from the paper here, in Figure D.1.

|          |                          |           |                        |                          |                         |
|----------|--------------------------|-----------|------------------------|--------------------------|-------------------------|
|          | VSM                      | BM25      | MLM <sub>jm</sub>      | MLM' <sub>dir</sub>      | MLM <sub>dir</sub>      |
| MAP      | .076                     | .079      | .084                   | .072                     | .066                    |
| TypesExp | »                        | »         | »                      | »                        | »                       |
| Types2X  | ≈                        | ≈         | ≈                      | ≈                        | ≈                       |
|          | SPUD                     | TF-IDFExp | MLM <sub>jm</sub> +RM3 | MLM' <sub>dir</sub> +RM3 | MLM <sub>dir</sub> +RM3 |
| MAP      | .090                     | .060      | .063                   | .051                     | .082                    |
| TypesExp | »                        | »         | »                      | »                        | »                       |
| Types2X  | ≈                        | ≈         | ≈                      | >                        | ≈                       |
|          | MLM <sub>dir</sub> +PRM2 | SPUD+RM3  | SPUD+PRM2              | Types2X                  | TypesExp                |
| MAP      | .061                     | .050      | .072                   | .094                     | .160                    |
| TypesExp | »                        | »         | »                      | >                        | –                       |
| Types2X  | ≈                        | >         | ≈                      | –                        | <                       |

Table D.1: Model MAP performance compared to TypesExp and Types2X models (expanded type dictionary).

|          |                          |           |                        |                          |                         |
|----------|--------------------------|-----------|------------------------|--------------------------|-------------------------|
|          | VSM                      | BM25      | MLM <sub>jm</sub>      | MLM' <sub>dir</sub>      | MLM <sub>dir</sub>      |
| MAP      | .076                     | .079      | .084                   | .072                     | .066                    |
| TypesExp | ≫                        | ≫         | ≫                      | ≫                        | ≫                       |
| Types2X  | ≈                        | ≈         | ≈                      | ≈                        | ≈                       |
|          | SPUD                     | TF-IDFExp | MLM <sub>jm</sub> +RM3 | MLM' <sub>dir</sub> +RM3 | MLM <sub>dir</sub> +RM3 |
| MAP      | .090                     | .060      | .063                   | .051                     | .082                    |
| TypesExp | ≫                        | ≫         | ≫                      | ≫                        | ≫                       |
| Types2X  | ≈                        | ≈         | ≈                      | >                        | ≈                       |
|          | MLM <sub>dir</sub> +PRM2 | SPUD+RM3  | SPUD+PRM2              | Types2X                  | TypesExp                |
| MAP      | .061                     | .050      | .072                   | <b>.094</b>              | <b>.150</b>             |
| TypesExp | ≫                        | ≫         | ≫                      | >                        | -                       |
| Types2X  | ≈                        | >         | ≈                      | -                        | <                       |

Table D.2: Model MAP performance compared to TypesExp and Types2X models (seed dictionary only).

## D.2 Section 7.2

### D.2.1 Subset and Statistics Definitions for Post-Hoc Analyses

| <b>Formula statistics</b>            |  |
|--------------------------------------|--|
| <b>Statistic</b>                     | <b>Description</b>   |
| Avg. No. Formulae                    | Average number of formulae in the subset queries or relevant documents   |
| <b>Word statistics</b>               |  |
| <b>Statistic</b>                     | <b>Description</b>   |
| Avg. No. Words                       | Average number of words in the subset queries or relevant documents  |
| <b>Type statistics</b>               |  |
| <b>Statistic</b>                     | <b>Description</b>   |
| Avg. No. Types                       | Average number of types in the subset queries or relevant documents  |
| <b>Typing statistics</b>             |  |
| <b>Statistic</b>                     | <b>Description</b>   |
| Avg. No. Positive Typings            | Average number of positive typing edges in the subset queries or relevant documents                                  |
| Avg. No. Negative Typings            | Average number of negative typing edges in the subset queries or relevant documents                                  |
| Avg. No. Typing Edges                | Average of all candidate typing edges in the partition queries or documents  |
| <b>Formula complexity statistics</b> |  |
| <b>Statistic</b>                     | <b>Description</b>   |
| Avg. total MathML Nodes              | Average of the total number of MathML nodes in the subset queries or relevant documents                              |
| Avg. Formula Complexity              | The average complexity of a formulae (per formula) in the subset queries or relevant document                        |
| Document total nodes                 | Average of the total number of MathML nodes in the subset (i.e., average complexity per query or relevant document). |

Table D.3: Complete set of statistics collected for each subset for profiling the performance of a pair of models. Type and Typing statistics are only used for post-hoc analysis in Section 7.3 (typed models).

## D.2.2 Subset $Q_F$

|            |      |        |          |          |         |           |
|------------|------|--------|----------|----------|---------|-----------|
| VSM+TM1-   | .053 | ≈      |          |          |         |           |
| VSM+TM2-   | .029 | ≪      | <        |          |         |           |
| BM25+UT    | .068 | ≈      | ≈        | ≫        |         |           |
| BM25+TM1-  | .082 | ≈      | ≈        | ≫        | ≈       |           |
| BM25+TM2-  | .062 | ≈      | ≈        | ≈        | ≈       | ≪         |
| <b>MAP</b> |      | .088   | .053     | .029     | .068    | .082      |
|            |      | VSM+UT | VSM+TM1- | VSM+TM2- | BM25+UT | BM25+TM1- |

Table D.4: Comparison between formula retrieval models on  $Q_F$  ( $\alpha$  tuned using DevSet;8Q).

|                          |            | BM25+TM1-     | BM25+TM2-      | BM25+UT        | VSM+TM1-       | VSM+TM2-       | VSM+UT        |
|--------------------------|------------|---------------|----------------|----------------|----------------|----------------|---------------|
| <b>Section 7.1 model</b> | <b>MAP</b> | .082          | .062           | .068           | .053           | .029           | .088          |
| VSM                      | .069       | -.014         | +.007          | +.001          | +.015          | <b>**+.040</b> | -.019         |
| BM25                     | .072       | -.010         | +.010          | +.004          | +.019          | <b>**+.043</b> | -.016         |
| MLM <sub>dir</sub>       | .076       | -.007         | +.014          | +.008          | +.022          | <b>*+.046</b>  | -.013         |
| MLM <sub>dir</sub> +PRM2 | .078       | -.005         | +.016          | +.010          | +.024          | <b>*+.049</b>  | -.010         |
| MLM <sub>dir</sub> +RM3  | .090       | +.008         | +.028          | +.022          | +.037          | <b>*+.061</b>  | +.002         |
| MLM <sub>jm</sub>        | .091       | +.009         | +.029          | +.023          | +.037          | <b>**+.062</b> | +.003         |
| MLM <sub>jm</sub> +RM3   | .074       | -.009         | +.012          | +.006          | +.020          | +.045          | -.014         |
| MLM' <sub>dir</sub>      | .075       | -.008         | +.013          | +.007          | +.021          | <b>*+.045</b>  | -.014         |
| MLM' <sub>dir</sub> +RM3 | .046       | -.036         | -.016          | -.022          | -.007          | +.017          | -.042         |
| SPUD                     | .098       | +.016         | +.037          | +.030          | +.045          | <b>**+.069</b> | +.010         |
| SPUD+PRM2                | .073       | -.010         | +.011          | +.005          | +.019          | <b>*+.044</b>  | -.015         |
| SPUD+RM3                 | .044       | -.038         | -.018          | -.024          | -.009          | +.015          | -.044         |
| TF-IDFExp                | .053       | -.029         | -.008          | -.015          | -.000          | +.024          | -.035         |
| Types2X                  | .071       | -.011         | +.009          | +.003          | +.018          | <b>*+.042</b>  | -.017         |
| Types2XExp               | .156       | <b>*+.073</b> | <b>**+.094</b> | <b>**+.088</b> | <b>**+.102</b> | <b>**+.127</b> | <b>*+.068</b> |

Table D.5: Comparison of formula retrieval models (parameters tuned on the DevSet;8Q) with the textual retrieval models from Section 7.1 on  $Q_F$ .

## D.2.3 Subset Q

| $\alpha$      | Textual model | No-Formula model | TM1-         | TM2-              | UT                |
|---------------|---------------|------------------|--------------|-------------------|-------------------|
| DevSet tuning | VSM           | .075             | .065 (-.010) | .048 <b>-.027</b> | .091 (+.016)      |
|               | BM25          | .072             | .084 (+.012) | .069 (-.003)      | .074 (+.002)      |
| f-t-t ratio   | VSM           | .075             | .086 (+.011) | .046 <b>-.029</b> | .079 <b>+.004</b> |
|               | BM25          | .072             | .085 (+.013) | .064 (-.008)      | .071 (-.001)      |

Table D.6: Performance (MAP) of untyped Formula Models on  $Q$  (160 Queries).

|            |      |        |          |         |           |           |
|------------|------|--------|----------|---------|-----------|-----------|
| VSM+TM1-   | .086 | ≈      |          |         |           |           |
| BM25+UT    | .071 | ≈      | ≈        |         |           |           |
| BM25+TM1-  | .085 | ≈      | ≈        | ≈       |           |           |
| BM25+TM2-  | .064 | ≈      | ≈        | ≈       | ≪         |           |
| VSM+TM2-   | .046 | ≪      | ≪        | ≈       | <         | ≈         |
| <b>MAP</b> |      | .079   | .086     | .071    | .085      | .064      |
|            |      | VSM+UT | VSM+TM1- | BM25+UT | BM25+TM1- | BM25+TM2- |

Table D.7: Comparison between formula retrieval models on **Q** ( $\alpha$  tuned using f-t-t).

|            |      |        |          |          |         |           |
|------------|------|--------|----------|----------|---------|-----------|
| VSM+TM1-   | .065 | ≈      |          |          |         |           |
| VSM+TM2-   | .048 | ≪      | <        |          |         |           |
| BM25+UT    | .074 | ≈      | ≈        | ≈        |         |           |
| BM25+TM1-  | .084 | ≈      | ≈        | ≈        | ≈       |           |
| BM25+TM2-  | .069 | ≈      | ≈        | ≈        | ≈       | ≪         |
| <b>MAP</b> |      | .091   | .065     | .048     | .074    | .084      |
|            |      | VSM+UT | VSM+TM1- | VSM+TM2- | BM25+UT | BM25+TM1- |

Table D.8: Comparison between formula retrieval models on **Q** ( $\alpha$  tuned using DevSet;13Q).

|                          |            | BM25+TM1-      | BM25+TM2-      | BM25+UT        | VSM+TM1-       | VSM+TM2-       | VSM+UT         |
|--------------------------|------------|----------------|----------------|----------------|----------------|----------------|----------------|
| <b>Section 7.1 model</b> | <b>MAP</b> | .084           | .069           | .074           | .065           | .048           | .091           |
| VSM                      | .076       | -.008          | +.007          | +.002          | +.011          | <b>*+.028</b>  | -.015          |
| BM25                     | .077       | -.007          | +.008          | +.003          | +.012          | <b>*+.029</b>  | -.014          |
| MLM <sub>dir</sub>       | .066       | -.018          | -.003          | -.008          | +.000          | +.018          | -.025          |
| MLM <sub>dir</sub> +PRM2 | .061       | -.023          | -.008          | -.012          | -.004          | +.013          | -.029          |
| MLM <sub>dir</sub> +RM3  | .082       | -.002          | +.013          | +.008          | +.016          | +.034          | -.009          |
| MLM <sub>jm</sub>        | .084       | +.000          | +.015          | +.011          | +.019          | +.036          | -.007          |
| MLM <sub>jm</sub> +RM3   | .063       | -.021          | -.006          | -.011          | -.003          | +.015          | -.028          |
| MLM' <sub>dir</sub>      | .072       | -.012          | +.003          | -.001          | +.007          | +.024          | -.019          |
| MLM' <sub>dir</sub> +RM3 | .051       | -.033          | -.018          | -.023          | -.015          | +.003          | -.040          |
| SPUD                     | .090       | +.006          | +.021          | +.017          | +.025          | +.042          | -.000          |
| SPUD+PRM2                | .072       | -.013          | +.002          | -.002          | +.006          | +.024          | -.019          |
| SPUD+RM3                 | .050       | -.034          | -.019          | -.024          | -.015          | +.002          | -.041          |
| TF-IDFExp                | .060       | -.024          | -.009          | -.013          | -.005          | +.012          | -.030          |
| Types2X                  | .070       | -.014          | +.001          | -.004          | +.004          | +.022          | -.021          |
| Types2XExp               | .173       | <b>**+.089</b> | <b>**+.104</b> | <b>**+.099</b> | <b>**+.107</b> | <b>**+.125</b> | <b>**+.082</b> |

Table D.9: Comparison of formula retrieval models (parameters tuned on the DevSet) with textual retrieval models from Section 7.1 on **Q**.

## D.2.4 Subset $Q_{FT}$

| $\alpha$      | Textual model | No-Formula model | TM1-         | TM2-         | UT           |
|---------------|---------------|------------------|--------------|--------------|--------------|
| DevSet tuning | VSM           | .065             | .055 (-.010) | .028 (-.037) | .088 (+.023) |
|               | BM25          | .067             | .085 (+.018) | .064 (-.003) | .066 (-.001) |
| f-t-t ratio   | VSM           | .065             | .082 (+.017) | .026 (-.039) | .070 (+.005) |
|               | BM25          | .067             | .087 (+.020) | .057 (-.010) | .065 (-.002) |

Table D.10: Performance (MAP) of untyped Formula Models on  $Q_{FT}$  (106 Queries): Tuning  $\alpha$ .

|            |      |        |          |         |           |           |
|------------|------|--------|----------|---------|-----------|-----------|
| VSM+TM1-   | .082 | ≈      |          |         |           |           |
| BM25+UT    | .065 | ≈      | ≈        |         |           |           |
| BM25+TM1-  | .087 | ≈      | ≈        | ≈       |           |           |
| BM25+TM2-  | .057 | ≈      | ≈        | ≈       | ≪         |           |
| VSM+TM2-   | .026 | ≪      | ≪        | <       | ≪         | ≈         |
| <b>MAP</b> |      | .070   | .082     | .065    | .087      | .057      |
|            |      | VSM+UT | VSM+TM1- | BM25+UT | BM25+TM1- | BM25+TM2- |

Table D.11: Comparison between formula retrieval models on  $Q_{FT}$  ( $\alpha$  tuned using f-t-t).

|            |      |        |          |          |         |           |
|------------|------|--------|----------|----------|---------|-----------|
| VSM+TM1-   | .055 | ≈      |          |          |         |           |
| VSM+TM2-   | .028 | ≪      | <        |          |         |           |
| BM25+UT    | .066 | ≈      | ≈        | ≫        |         |           |
| BM25+TM1-  | .085 | ≈      | ≈        | >        | ≈       |           |
| BM25+TM2-  | .064 | ≈      | ≈        | ≈        | ≈       | ≪         |
| <b>MAP</b> |      | .088   | .055     | .028     | .066    | .085      |
|            |      | VSM+UT | VSM+TM1- | VSM+TM2- | BM25+UT | BM25+TM1- |

Table D.12: Comparison between formula retrieval models on  $Q_{FT}$  ( $\alpha$  tuned using DevSet;8Q).

|                          |            | BM25+TM1-     | BM25+TM2-      | BM25+UT        | VSM+TM1-      | VSM+TM2-       | VSM+UT         |
|--------------------------|------------|---------------|----------------|----------------|---------------|----------------|----------------|
| <b>Section 7.1 model</b> | <b>MAP</b> | .087          | .057           | .065           | .082          | .026           | .070           |
| VSM                      | .072       | -.014         | +.016          | +.008          | -.010         | <b>**+.047</b> | +.003          |
| BM25                     | .071       | -.016         | +.014          | +.006          | -.012         | <b>**+.045</b> | +.001          |
| MLM <sub>dir</sub>       | .070       | -.017         | +.013          | +.005          | -.013         | <b>*+.044</b>  | +.000          |
| MLM <sub>dir</sub> +PRM2 | .075       | -.012         | +.019          | +.011          | -.007         | <b>*+.049</b>  | +.006          |
| MLM <sub>dir</sub> +RM3  | .081       | -.006         | +.025          | +.017          | -.001         | <b>*+.055</b>  | +.012          |
| MLM <sub>jm</sub>        | .088       | +.001         | +.031          | +.023          | +.005         | <b>**+.062</b> | +.018          |
| MLM <sub>jm</sub> +RM3   | .066       | -.021         | +.009          | +.001          | -.016         | +.040          | -.004          |
| MLM' <sub>dir</sub>      | .070       | -.017         | +.013          | +.005          | -.013         | <b>*+.044</b>  | +.000          |
| MLM' <sub>dir</sub> +RM3 | .036       | <b>*-.051</b> | -.020          | -.028          | -.046         | +.010          | -.033          |
| SPUD                     | .095       | +.008         | +.038          | +.030          | +.012         | <b>**+.069</b> | +.025          |
| SPUD+PRM2                | .070       | -.017         | +.014          | +.006          | -.012         | +.044          | +.001          |
| SPUD+RM3                 | .035       | -.052         | -.021          | -.029          | <b>*-.047</b> | +.009          | -.034          |
| TF-IDFExp                | .047       | -.040         | -.009          | -.017          | -.035         | +.021          | -.022          |
| Types2X                  | .073       | -.014         | +.016          | +.008          | -.009         | <b>**+.047</b> | +.003          |
| Types2XExp               | .154       | <b>*+.067</b> | <b>**+.098</b> | <b>**+.090</b> | <b>*+.072</b> | <b>**+.128</b> | <b>**+.085</b> |

Table D.13: Comparison of formula retrieval models (parameters tuned on f-t-t) with textual retrieval models from Section 7.1 on  $\mathbf{Q}_{FT}$  (106 queries).

|                          |            | BM25+TM1-     | BM25+TM2-      | BM25+UT       | VSM+TM1-       | VSM+TM2-       | VSM+UT        |
|--------------------------|------------|---------------|----------------|---------------|----------------|----------------|---------------|
| <b>Section 7.1 model</b> | <b>MAP</b> | .085          | .064           | .066          | .055           | .028           | .088          |
| VSM                      | .072       | -.012         | +.009          | +.006         | +.018          | <b>**+.044</b> | -.015         |
| BM25                     | .071       | -.014         | +.007          | +.005         | +.016          | <b>**+.043</b> | -.017         |
| MLM <sub>dir</sub>       | .070       | -.015         | +.006          | +.004         | +.015          | +.042          | -.018         |
| MLM <sub>dir</sub> +PRM2 | .075       | -.010         | +.011          | +.009         | +.020          | <b>*+.047</b>  | -.013         |
| MLM <sub>dir</sub> +RM3  | .081       | -.003         | +.017          | +.015         | +.026          | <b>*+.053</b>  | -.006         |
| MLM <sub>jm</sub>        | .088       | +.003         | +.024          | +.022         | +.033          | <b>*+.060</b>  | +.000         |
| MLM <sub>jm</sub> +RM3   | .066       | -.019         | +.002          | -.000         | +.011          | +.038          | -.022         |
| MLM' <sub>dir</sub>      | .070       | -.015         | +.006          | +.004         | +.015          | <b>*+.042</b>  | -.018         |
| MLM' <sub>dir</sub> +RM3 | .036       | <b>*-.049</b> | -.028          | -.030         | -.019          | +.008          | <b>*-.052</b> |
| SPUD                     | .095       | +.010         | +.031          | +.029         | +.040          | <b>**+.067</b> | +.007         |
| SPUD+PRM2                | .070       | -.015         | +.006          | +.004         | +.015          | +.042          | -.017         |
| SPUD+RM3                 | .035       | -.050         | -.029          | -.031         | -.020          | +.007          | <b>*-.053</b> |
| TF-IDFExp                | .047       | -.038         | -.017          | -.019         | -.008          | +.019          | -.041         |
| Types2X                  | .073       | -.012         | +.009          | +.007         | +.018          | <b>*+.045</b>  | -.015         |
| Types2XExp               | .154       | <b>*+.070</b> | <b>**+.090</b> | <b>*+.088</b> | <b>**+.099</b> | <b>**+.126</b> | <b>*+.067</b> |

Table D.14: Comparison of formula retrieval models (parameters tuned on DevSet;8Q) with textual retrieval models from Section 7.1 on  $\mathbf{Q}_{FT}$  (106 queries).

## D.2.5 Section 7.2, Post-hoc Analyses

| Num.       | Query | SubQ |
|------------|-------|------|------------|-------|------|------------|-------|------|------------|-------|------|
| <b>1</b>   | 237   | 3    | <b>2</b>   | 232   | 1    | <b>3</b>   | 112   | 1    | <b>4</b>   | 352   | 1    |
| <b>5</b>   | 18    | 1    | <b>6</b>   | 56    | 1    | <b>7</b>   | 36    | 2    | <b>8</b>   | 250   | 3    |
| <b>9</b>   | 205   | 1    | <b>10</b>  | 108   | 1    | <b>11</b>  | 135   | 2    | <b>12</b>  | 180   | 1    |
| <b>13</b>  | 135   | 3    | <b>14</b>  | 92    | 1    | <b>15</b>  | 59    | 1    | <b>16</b>  | 121   | 1    |
| <b>17</b>  | 21    | 1    | <b>18</b>  | 242   | 1    | <b>19</b>  | 211   | 1    | <b>20</b>  | 56    | 2    |
| <b>21</b>  | 246   | 2    | <b>22</b>  | 311   | 2    | <b>23</b>  | 167   | 1    | <b>24</b>  | 352   | 3    |
| <b>25</b>  | 335   | 2    | <b>26</b>  | 26    | 2    | <b>27</b>  | 25    | 1    | <b>28</b>  | 120   | 1    |
| <b>29</b>  | 228   | 2    | <b>30</b>  | 143   | 1    | <b>31</b>  | 39    | 1    | <b>32</b>  | 8     | 1    |
| <b>33</b>  | 301   | 1    | <b>34</b>  | 31    | 2    | <b>35</b>  | 200   | 1    | <b>36</b>  | 91    | 1    |
| <b>37</b>  | 167   | 2    | <b>38</b>  | 189   | 2    | <b>39</b>  | 222   | 1    | <b>40</b>  | 220   | 1    |
| <b>41</b>  | 180   | 2    | <b>42</b>  | 264   | 1    | <b>43</b>  | 12    | 1    | <b>44</b>  | 301   | 2    |
| <b>45</b>  | 311   | 1    | <b>46</b>  | 103   | 2    | <b>47</b>  | 287   | 1    | <b>48</b>  | 69    | 1    |
| <b>49</b>  | 82    | 1    | <b>50</b>  | 76    | 1    | <b>51</b>  | 335   | 1    | <b>52</b>  | 134   | 1    |
| <b>53</b>  | 228   | 3    | <b>54</b>  | 250   | 1    | <b>55</b>  | 228   | 1    | <b>56</b>  | 49    | 1    |
| <b>57</b>  | 287   | 2    | <b>58</b>  | 310   | 1    | <b>59</b>  | 117   | 1    | <b>60</b>  | 88    | 1    |
| <b>61</b>  | 32    | 2    | <b>62</b>  | 317   | 1    | <b>63</b>  | 175   | 1    | <b>64</b>  | 29    | 1    |
| <b>65</b>  | 27    | 1    | <b>66</b>  | 91    | 3    | <b>67</b>  | 187   | 1    | <b>68</b>  | 83    | 1    |
| <b>69</b>  | 31    | 3    | <b>70</b>  | 189   | 1    | <b>71</b>  | 246   | 1    | <b>72</b>  | 235   | 1    |
| <b>73</b>  | 218   | 1    | <b>74</b>  | 82    | 2    | <b>75</b>  | 268   | 1    | <b>76</b>  | 114   | 1    |
| <b>77</b>  | 35    | 1    | <b>78</b>  | 322   | 1    | <b>79</b>  | 354   | 1    | <b>80</b>  | 36    | 1    |
| <b>81</b>  | 148   | 1    | <b>82</b>  | 31    | 1    | <b>83</b>  | 28    | 2    | <b>84</b>  | 165   | 1    |
| <b>85</b>  | 334   | 1    | <b>86</b>  | 72    | 2    | <b>87</b>  | 309   | 1    | <b>88</b>  | 286   | 3    |
| <b>89</b>  | 301   | 3    | <b>90</b>  | 117   | 2    | <b>91</b>  | 19    | 1    | <b>92</b>  | 181   | 1    |
| <b>93</b>  | 258   | 3    | <b>94</b>  | 176   | 2    | <b>95</b>  | 352   | 2    | <b>96</b>  | 152   | 1    |
| <b>97</b>  | 196   | 1    | <b>98</b>  | 248   | 1    | <b>99</b>  | 62    | 1    | <b>100</b> | 135   | 1    |
| <b>101</b> | 73    | 2    | <b>102</b> | 348   | 1    | <b>103</b> | 253   | 1    | <b>104</b> | 231   | 1    |
| <b>105</b> | 221   | 1    | <b>106</b> | 76    | 2    | <b>107</b> | 13    | 1    | <b>108</b> | 131   | 1    |
| <b>109</b> | 13    | 2    | <b>110</b> | 214   | 1    | <b>111</b> | 272   | 1    | <b>112</b> | 34    | 1    |
| <b>113</b> | 345   | 1    | <b>114</b> | 356   | 1    | <b>115</b> | 87    | 1    | <b>116</b> | 87    | 2    |

Table D.15: Mapping of query IDs to CUMTC queries as shown on the x-axis of the difference in reciprocal rank plots in Section 7.2.2.

## D.2.5.1 Post-hoc Analysis 2

| Collection Formula-to-Token Estimation of $\alpha$ |                               |                    |                      |        |                   |         |                         |       |  |
|--|-------------------------------|--------------------|----------------------|--------|-------------------|---------|-------------------------|-------|--|
| Subset ID  | Subset label                  | Relevant documents | Avg. Formula Density |        | Avg. Word Density |         | Avg. Formula Complexity |       |  |
|  |                               |                    | Q                    | D      | Q                 | D       | Q                       | D     |  |
| (0)  | $Q_{FT}$                      | 131                | 9.98                 | 736.13 | 117.64            | 4726.14 | 15.54                   | 24.45 |  |
| (2)  | $B \setminus A$               | 11                 | 6.27                 | 661.0  | 132.0             | 4598.18 | 17.13                   | 20.7  |  |
| (3)  | $A \cap B$                    | 120                | 10.34                | 743.02 | 116.31            | 4737.87 | 15.31                   | 24.8  |  |
| (5)  | $A \cap B, rank(A) > rank(B)$ | 30                 | 13.71                | 461.9  | 131.54            | 2698.6  | 16.12                   | 22.19 |  |
| (6)  | $A \cap B, rank(B) > rank(A)$ | 71                 | 9.61                 | 823.14 | 109.14            | 5324.72 | 15.15                   | 27.08 |  |
| (7)  | $A \cap B, rank(B) = rank(A)$ | 19                 | 7.26                 | 887.47 | 108.11            | 5764.79 | 13.66                   | 20.38 |  |

| Development Set Estimation of $\alpha$ (8Q) |                               |                    |                      |         |                   |         |                         |       |  |
|---|-------------------------------|--------------------|----------------------|---------|-------------------|---------|-------------------------|-------|--|
| Subset ID                                   | Subset label                  | Relevant documents | Avg. Formula Density |         | Avg. Word Density |         | Avg. Formula Complexity |       |  |
|   |                               |                    | Q                    | D       | Q                 | D       | Q                       | D     |  |
| (0)   | $Q_{FT}$                      | 131                | 9.98                 | 736.13  | 117.64            | 4726.14 | 15.54                   | 24.45 |  |
| (2)   | $B \setminus A$               | 11                 | 6.27                 | 661.0   | 132.0             | 4598.18 | 17.13                   | 20.7  |  |
| (3)   | $A \cap B$                    | 120                | 10.34                | 743.02  | 116.31            | 4737.87 | 15.31                   | 24.8  |  |
| (5)   | $A \cap B, rank(A) > rank(B)$ | 54                 | 11.24                | 632.28  | 125.53            | 4229.78 | 14.65                   | 21.9  |  |
| (6)   | $A \cap B, rank(B) > rank(A)$ | 63                 | 9.0                  | 816.35  | 103.86            | 5026.63 | 15.66                   | 27.6  |  |
| (7)   | $A \cap B, rank(B) = rank(A)$ | 3                  | 14.0                 | 1196.33 | 122.67            | 7819.33 | 12.14                   | 18.22 |  |

Table D.16: Subset labels and statistics for investigating the behaviour of VSM (model A) and VSM+UT (model B) on  $Q_F$ : Top: model B tuned using f-t-t method, Bottom: model B tuned using DevSet.

## D.3 Section 7.3

### D.3.1 Significance Tests between All Models based on Types2ExExp

|                    |      |      |      |      |      |                    |                    |                    |                    |       |
|--------------------|------|------|------|------|------|--------------------|--------------------|--------------------|--------------------|-------|
| +TT                | .130 | ≪    |      |      |      |                    |                    |                    |                    |       |
| +TM1               | .140 | ≈    | ≈    |      |      |                    |                    |                    |                    |       |
| +TM2               | .142 | ≈    | ≈    | ≈    |      |                    |                    |                    |                    |       |
| +TM1 <sub>JD</sub> | .150 | ≈    | ≈    | ≈    | ≈    |                    |                    |                    |                    |       |
| +TM2 <sub>JD</sub> | .096 | ≪    | ≪    | ≪    | ≪    | ≪                  |                    |                    |                    |       |
| +TM1 <sub>JE</sub> | .151 | ≈    | ≈    | ≈    | ≈    | ≈                  | ≫                  |                    |                    |       |
| +TM2 <sub>JE</sub> | .106 | ≪    | <    | ≪    | ≪    | ≪                  | ≫                  | ≪                  |                    |       |
| +TM1-              | .147 | ≈    | ≈    | ≈    | ≈    | ≪                  | ≫                  | ≪                  | ≫                  |       |
| +TM2-              | .069 | ≪    | ≪    | ≪    | ≪    | ≪                  | ≪                  | ≪                  | ≪                  | ≪     |
| <b>MAP</b>         |      | .133 | .130 | .140 | .142 | .150               | .096               | .151               | .106               | .147  |
|                    |      | +UT  | +TT  | +TM1 | +TM2 | +TM1 <sub>JD</sub> | +TM2 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2 <sub>JE</sub> | +TM1- |

Table D.17: Comparison between typed retrieval models on  $Q_{FT}$  ( $\alpha$  tuned using f-t-t).

|                    |      |      |      |      |      |                    |                    |                    |                    |       |   |
|--------------------|------|------|------|------|------|--------------------|--------------------|--------------------|--------------------|-------|---|
| +TT                | .130 | ≈    |      |      |      |                    |                    |                    |                    |       |   |
| +TM1               | .132 | ≈    | ≈    |      |      |                    |                    |                    |                    |       |   |
| +TM2               | .138 | ≈    | ≈    | ≈    |      |                    |                    |                    |                    |       |   |
| +TM1 <sub>JD</sub> | .148 | ≈    | ≈    | ≈    | ≈    |                    |                    |                    |                    |       |   |
| +TM2 <sub>JD</sub> | .111 | ≈    | ≈    | ≪    | <    | <                  |                    |                    |                    |       |   |
| +TM1 <sub>JE</sub> | .138 | ≈    | ≈    | ≈    | ≈    | ≈                  | ≈                  | ≫                  |                    |       |   |
| +TM2 <sub>JE</sub> | .121 | ≈    | ≈    | ≈    | ≈    | ≈                  | ≈                  | ≫                  | ≈                  |       |   |
| +TM1-              | .141 | ≈    | ≈    | ≈    | ≈    | ≈                  | ≈                  | ≈                  | ≈                  | ≈     |   |
| +TM2-              | .079 | ≈    | ≪    | ≪    | ≪    | ≪                  | ≪                  | ≪                  | ≪                  | ≪     | ≪ |
| MAP                |      | .109 | .130 | .132 | .138 | .148               | .111               | .138               | .121               | .141  |   |
|                    |      | +UT  | +TT  | +TM1 | +TM2 | +TM1 <sub>JD</sub> | +TM2 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2 <sub>JE</sub> | +TM1- |   |

Table D.18: Comparison between typed retrieval models on  $Q_{FT}$  ( $\alpha$  tuned using DevSet).

### D.3.2 Typed Retrieval Results on $Q$ , $Q_F$ and $Q_{FT}$ (DevSet)

#### D.3.2.1 DevSet Results on $Q_{FT}$

|                          |              | Formula Model |         |              |       |       |       |              |                    |                    |                    |                    |
|--------------------------|--------------|---------------|---------|--------------|-------|-------|-------|--------------|--------------------|--------------------|--------------------|--------------------|
|                          |              | None          | Untyped |              |       | Typed |       |              |                    |                    |                    |                    |
| Textual model (baseline) |              |               | +TM1-   | +TM2-        | +UT   | +TT   | +TM1  | +TM2         | +TM1 <sub>JD</sub> | +TM2 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2 <sub>JE</sub> |
| VSM                      | MAP          | .065          | .055    | .028         | .088  | .082  | .056  | .030         | .052               | .036               | .057               | .042               |
|                          | MAP $\Delta$ |               | -.010   | <b>-.037</b> | +023  | +017  | -.009 | <b>-.035</b> | -.013              | <b>-.029</b>       | -.008              | <b>-.023</b>       |
| BM25                     | MAP          | .067          | .085    | .064         | .066  | .067  | .085  | .084         | .087               | .065               | .085               | .066               |
|                          | MAP $\Delta$ |               | +018    | -.003        | -.001 | +0    | +018  | +017         | +020               | -.002              | +018               | -.001              |
| Types                    | MAP          | .039          | .048    | .029         | .039  | .040  | .039  | .036         | .039               | .034               | .039               | .034               |
|                          | MAP $\Delta$ |               | +009    | <b>-.010</b> | +0    | +001  | +0    | -.003        | +0                 | -.005              | +0                 | -.005              |
| Types2X                  | MAP          | .068          | .087    | .037         | .056  | .065  | .088  | .065         | .085               | .050               | .090               | .060               |
|                          | MAP $\Delta$ |               | +019    | <b>-.031</b> | -.012 | -.003 | +020  | -.003        | +017               | -.018              | +022               | -.008              |
| Types2XExp               | MAP          | .131          | .141    | .079         | .109  | .130  | .132  | .138         | .148               | .111               | .138               | .121               |
|                          | MAP $\Delta$ |               | +010    | <b>-.052</b> | -.022 | -.001 | +001  | +007         | +017               | <b>-.020</b>       | +007               | -.010              |

Table D.19: Performance of untyped and typed (joint) formula retrieval models on  $Q_{FT}$  (106 queries), including differences from respective text-only baseline model (bold if significant). Tuning using the DevSet method.

### D.3.2.2 Results on Q

| Textual model (baseline) |              | None | Formula Model |              |              |       |       |       |                    |                    |                    |                    |
|--------------------------|--------------|------|---------------|--------------|--------------|-------|-------|-------|--------------------|--------------------|--------------------|--------------------|
|                          |              |      | Untyped       |              |              | Typed |       |       |                    |                    |                    |                    |
|                          |              |      | +TM1-         | +TM2-        | +UT          | +TT   | +TM1  | +TM2  | +TM1 <sub>JD</sub> | +TM2 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2 <sub>JE</sub> |
| VSM                      | MAP          | .075 | .086          | .046         | .079         | .078  | .082  | .079  | .088               | .053               | .086               | .055               |
|                          | MAP $\Delta$ |      | +.011         | <b>-.029</b> | <b>+.004</b> | +.003 | +.007 | +.004 | +.013              | <b>-.022</b>       | +.011              | <b>-.020</b>       |
| BM25                     | MAP          | .072 | .085          | .064         | .071         | .070  | .084  | .083  | .084               | .070               | .084               | .070               |
|                          | MAP $\Delta$ |      | +.013         | -.008        | -.001        | -.002 | +.012 | +.011 | +.012              | -.002              | +.012              | -.002              |
| Types                    | MAP          | .063 | .070          | .050         | .063         | .064  | .063  | .064  | .070               | .054               | .070               | .054               |
|                          | MAP $\Delta$ |      | +.007         | <b>-.013</b> | +.000        | +.001 | +.000 | +.001 | +.007              | <b>-.009</b>       | +.007              | <b>-.009</b>       |
| Types2X                  | MAP          | .077 | .091          | .060         | .074         | .073  | .082  | .086  | .083               | .065               | .086               | .072               |
|                          | MAP $\Delta$ |      | +.014         | <b>-.017</b> | -.003        | -.004 | +.005 | +.009 | +.006              | -.012              | +.009              | -.005              |
| Types2XExp               | MAP          | .159 | .168          | .114         | .161         | .157  | .164  | .163  | .171               | .132               | .171               | .138               |
|                          | MAP $\Delta$ |      | +.009         | <b>-.045</b> | <b>+.002</b> | -.002 | +.005 | +.004 | +.012              | <b>-.027</b>       | +.012              | <b>-.021</b>       |

Table D.20: Performance of untyped and typed (joint) formula retrieval models on Q (160 queries), including differences from respective text-only baseline model (bold if significant). Tuning using the f-t-t method.

|                    |      |      |      |      |      |                    |                    |                    |                    |       |  |   |
|--------------------|------|------|------|------|------|--------------------|--------------------|--------------------|--------------------|-------|--|---|
| +TT                | .157 | «    |      |      |      |                    |                    |                    |                    |       |  |   |
| +TM1               | .164 | ≈    | >    |      |      |                    |                    |                    |                    |       |  |   |
| +TM2               | .163 | ≈    | ≈    | ≈    |      |                    |                    |                    |                    |       |  |   |
| +TM1 <sub>JD</sub> | .171 | ≈    | ≈    | ≈    | ≈    |                    |                    |                    |                    |       |  |   |
| +TM2 <sub>JD</sub> | .132 | «    | «    | «    | «    | «                  |                    |                    |                    |       |  |   |
| +TM1 <sub>JE</sub> | .171 | ≈    | ≈    | ≈    | >    | ≈                  | »                  |                    |                    |       |  |   |
| +TM2 <sub>JE</sub> | .138 | «    | <    | «    | «    | «                  | »                  | «                  |                    |       |  |   |
| +TM1-              | .168 | ≈    | ≈    | ≈    | ≈    | «                  | »                  | «                  | »                  |       |  |   |
| +TM2-              | .114 | «    | «    | «    | «    | «                  | «                  | «                  | «                  | «     |  | « |
| <b>MAP</b>         |      | .161 | .157 | .164 | .163 | .171               | .132               | .171               | .138               | .168  |  |   |
|                    |      | +UT  | +TT  | +TM1 | +TM2 | +TM1 <sub>JD</sub> | +TM2 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2 <sub>JE</sub> | +TM1- |  |   |

Table D.21: Comparison between typed retrieval models on Q ( $\alpha$  tuned using f-t-t).

|                          |              | Formula Model |         |              |        |        |        |              |                    |                    |                    |                    |
|--------------------------|--------------|---------------|---------|--------------|--------|--------|--------|--------------|--------------------|--------------------|--------------------|--------------------|
|                          |              | None          | Untyped |              |        | Typed  |        |              |                    |                    |                    |                    |
| Textual model (baseline) |              |               | +TM1-   | +TM2-        | +UT    | +TT    | +TM1   | +TM2         | +TM1 <sub>JD</sub> | +TM2 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2 <sub>JE</sub> |
| VSM                      | MAP          | .075          | .065    | .048         | .091   | .085   | .067   | .049         | .063               | .054               | .067               | .057               |
|                          | MAP $\Delta$ |               | -.010   | <b>-.027</b> | +0.016 | +0.010 | -.008  | <b>-.026</b> | -.012              | <b>-.021</b>       | -.008              | <b>-.018</b>       |
| BM25                     | MAP          | .072          | .084    | .069         | .074   | .072   | .084   | .083         | .085               | .070               | .084               | .070               |
|                          | MAP $\Delta$ |               | +0.012  | -.003        | +0.002 | +0.000 | +0.012 | +0.011       | +0.013             | -.002              | +0.012             | -.002              |
| Types                    | MAP          | .063          | .069    | .051         | .064   | .064   | .063   | .058         | .063               | .055               | .063               | .055               |
|                          | MAP $\Delta$ |               | +0.006  | <b>-.012</b> | +0.001 | +0.001 | +0.000 | -.005        | +0.000             | <b>-.008</b>       | +0.000             | <b>-.008</b>       |
| Types2X                  | MAP          | .077          | .089    | .054         | .079   | .076   | .090   | .074         | .088               | .064               | .091               | .071               |
|                          | MAP $\Delta$ |               | +0.012  | <b>-.023</b> | +0.002 | -.001  | +0.013 | -.003        | +0.011             | <b>-.013</b>       | +0.014             | -.006              |
| Types2XExp               | MAP          | .159          | .163    | .120         | .153   | .158   | .160   | .161         | .170               | .142               | .163               | .149               |
|                          | MAP $\Delta$ |               | +0.004  | <b>-.039</b> | -.006  | -.001  | +0.001 | +0.002       | +0.011             | <b>-.017</b>       | +0.004             | -.010              |

Table D.22: Performance of untyped and typed (joint) formula retrieval models on Q (160 queries), including differences from respective text-only baseline model (bold if significant). Tuning using the DevSet.

|                    |      |      |      |      |      |                    |                    |                    |                    |       |   |
|--------------------|------|------|------|------|------|--------------------|--------------------|--------------------|--------------------|-------|---|
| +TT                | .158 | ≈    |      |      |      |                    |                    |                    |                    |       |   |
| +TM1               | .160 | ≈    | ≈    |      |      |                    |                    |                    |                    |       |   |
| +TM2               | .161 | ≈    | ≈    | ≈    |      |                    |                    |                    |                    |       |   |
| +TM1 <sub>JD</sub> | .170 | ≈    | ≈    | ≈    | ≈    |                    |                    |                    |                    |       |   |
| +TM2 <sub>JD</sub> | .142 | ≈    | <    | ≪    | <    | ≪                  |                    |                    |                    |       |   |
| +TM1 <sub>JE</sub> | .163 | ≈    | ≈    | ≈    | ≈    | ≈                  | ≫                  |                    |                    |       |   |
| +TM2 <sub>JE</sub> | .149 | ≈    | ≈    | ≈    | <    | <                  | ≫                  | ≈                  |                    |       |   |
| +TM1-              | .163 | ≈    | ≈    | ≈    | ≈    | ≪                  | >                  | ≈                  | ≈                  |       |   |
| +TM2-              | .120 | ≈    | ≪    | ≪    | ≪    | ≪                  | ≪                  | ≪                  | ≪                  | ≪     | ≪ |
| <b>MAP</b>         |      | .153 | .158 | .160 | .161 | .170               | .142               | .163               | .149               | .163  |   |
|                    |      | +UT  | +TT  | +TM1 | +TM2 | +TM1 <sub>JD</sub> | +TM2 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2 <sub>JE</sub> | +TM1- |   |

Table D.23: Comparison between typed retrieval models on Q ( $\alpha$  tuned using DevSet).

### D.3.2.3 Results on $Q_F$

| Textual model (baseline) |              | None | Formula Model |              |              |        |        |        |                    |                    |                    |                    |
|--------------------------|--------------|------|---------------|--------------|--------------|--------|--------|--------|--------------------|--------------------|--------------------|--------------------|
|                          |              |      | Untyped       |              |              | Typed  |        |        |                    |                    |                    |                    |
|                          |              |      | +TM1-         | +TM2-        | +UT          | +TT    | +TM1   | +TM2   | +TM1 <sub>JD</sub> | +TM2 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2 <sub>JE</sub> |
| VSM                      | MAP          | .067 | .081          | .027         | .071         | .070   | .076   | .072   | .084               | .037               | .082               | .039               |
|                          | MAP $\Delta$ |      | +0.014        | <b>-.040</b> | <b>+.004</b> | +0.003 | +0.009 | +0.005 | +0.017             | <b>-.030</b>       | +0.015             | <b>-.028</b>       |
| BM25                     | MAP          | .066 | .084          | .055         | .065         | .063   | .082   | .081   | .082               | .062               | .082               | .063               |
|                          | MAP $\Delta$ |      | +0.018        | -0.011       | -0.001       | -0.003 | +0.016 | +0.015 | +0.016             | -0.004             | +0.016             | -0.003             |
| Types                    | MAP          | .044 | .054          | .026         | .044         | .045   | .044   | .045   | .054               | .032               | .054               | .032               |
|                          | MAP $\Delta$ |      | +0.010        | <b>-.018</b> | +0.000       | +0.001 | +0.000 | +0.001 | +0.010             | <b>-.012</b>       | +0.010             | <b>-.012</b>       |
| Types2X                  | MAP          | .065 | .084          | .042         | .061         | .059   | .073   | .078   | .074               | .049               | .078               | .058               |
|                          | MAP $\Delta$ |      | +0.019        | <b>-.023</b> | -0.004       | -0.006 | +0.008 | +0.013 | +0.009             | -0.016             | +0.013             | -0.007             |
| Types2XExp               | MAP          | .127 | .139          | .065         | .129         | .124   | .135   | .133   | .144               | .090               | .144               | .099               |
|                          | MAP $\Delta$ |      | +0.012        | <b>-.062</b> | <b>+.002</b> | -0.003 | +0.008 | +0.006 | +0.017             | <b>-.037</b>       | +0.017             | <b>-.028</b>       |

Table D.24: Performance of untyped and typed (joint) formula retrieval models on  $Q_F$  (116 queries), including differences from respective text-only baseline model (bold if significant). Tuning using the f-t-t method.

|                    |      |      |      |      |      |                    |                    |                    |                    |       |   |   |
|--------------------|------|------|------|------|------|--------------------|--------------------|--------------------|--------------------|-------|---|---|
| +TT                | .124 | «    |      |      |      |                    |                    |                    |                    |       |   |   |
| +TM1               | .135 | ≈    | ≈    |      |      |                    |                    |                    |                    |       |   |   |
| +TM2               | .133 | ≈    | ≈    | ≈    |      |                    |                    |                    |                    |       |   |   |
| +TM1 <sub>JD</sub> | .144 | ≈    | ≈    | ≈    | ≈    |                    |                    |                    |                    |       |   |   |
| +TM2 <sub>JD</sub> | .090 | «    | «    | «    | «    | «                  |                    |                    |                    |       |   |   |
| +TM1 <sub>JE</sub> | .144 | ≈    | ≈    | ≈    | >    | ≈                  | »                  |                    |                    |       |   |   |
| +TM2 <sub>JE</sub> | .099 | «    | <    | «    | «    | «                  | »                  | «                  |                    |       |   |   |
| +TM1-              | .139 | ≈    | ≈    | ≈    | ≈    | «                  | »                  | «                  | »                  |       |   |   |
| +TM2-              | .065 | «    | «    | «    | «    | «                  | «                  | «                  | «                  | «     | « | « |
| <b>MAP</b>         |      | .129 | .124 | .135 | .133 | .144               | .090               | .144               | .099               | .139  |   |   |
|                    |      | +UT  | +TT  | +TM1 | +TM2 | +TM1 <sub>JD</sub> | +TM2 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2 <sub>JE</sub> | +TM1- |   |   |

Table D.25: Comparison between typed retrieval models on  $Q_F$  ( $\alpha$  tuned using f-t-t).

|                          |              | Formula Model |         |              |        |        |        |              |                    |                    |                    |                    |
|--------------------------|--------------|---------------|---------|--------------|--------|--------|--------|--------------|--------------------|--------------------|--------------------|--------------------|
|                          |              | None          | Untyped |              |        | Typed  |        |              |                    |                    |                    |                    |
| Textual model (baseline) |              |               | +TM1-   | +TM2-        | +UT    | +TT    | +TM1   | +TM2         | +TM1 <sub>JD</sub> | +TM2 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2 <sub>JE</sub> |
| VSM                      | MAP          | .067          | .053    | .029         | .088   | .080   | .055   | .031         | .051               | .037               | .056               | .042               |
|                          | MAP $\Delta$ |               | -.014   | <b>-.038</b> | +0.021 | +0.013 | -.012  | <b>-.036</b> | -.016              | <b>-.030</b>       | -.011              | <b>-.025</b>       |
| BM25                     | MAP          | .066          | .082    | .062         | .068   | .065   | .082   | .081         | .084               | .063               | .083               | .063               |
|                          | MAP $\Delta$ |               | +0.016  | -.004        | +0.002 | -.001  | +0.016 | +0.015       | +0.018             | -.003              | +0.017             | -.003              |
| Types                    | MAP          | .044          | .052    | .028         | .046   | .046   | .044   | .037         | .044               | .032               | .044               | .033               |
|                          | MAP $\Delta$ |               | +0.008  | <b>-.016</b> | +0.002 | +0.002 | +0.000 | -.007        | +0.000             | <b>-.012</b>       | +0.000             | <b>-.011</b>       |
| Types2X                  | MAP          | .065          | .082    | .035         | .068   | .064   | .083   | .062         | .081               | .048               | .085               | .057               |
|                          | MAP $\Delta$ |               | +0.017  | <b>-.030</b> | +0.003 | -.001  | +0.018 | -.003        | +0.016             | <b>-.017</b>       | +0.020             | -.008              |
| Types2XExp               | MAP          | .127          | .132    | .074         | .119   | .126   | .128   | .130         | .142               | .104               | .133               | .113               |
|                          | MAP $\Delta$ |               | +0.005  | <b>-.053</b> | -.008  | -.001  | +0.001 | +0.003       | +0.015             | <b>-.023</b>       | +0.006             | -.014              |

Table D.26: Performance of untyped and typed (joint) formula retrieval models on  $Q_F$  (116 queries), including differences from respective text-only baseline model (bold if significant). Tuning using the DevSet.

|                    |      |      |      |      |      |                    |                    |                    |                    |       |   |
|--------------------|------|------|------|------|------|--------------------|--------------------|--------------------|--------------------|-------|---|
| +TT                | .126 | ≈    |      |      |      |                    |                    |                    |                    |       |   |
| +TM1               | .128 | ≈    | ≈    |      |      |                    |                    |                    |                    |       |   |
| +TM2               | .130 | ≈    | ≈    | ≈    |      |                    |                    |                    |                    |       |   |
| +TM1 <sub>JD</sub> | .142 | ≈    | ≈    | ≈    | ≈    |                    |                    |                    |                    |       |   |
| +TM2 <sub>JD</sub> | .104 | ≈    | <    | ≪    | <    | ≪                  |                    |                    |                    |       |   |
| +TM1 <sub>JE</sub> | .133 | ≈    | ≈    | ≈    | ≈    | ≈                  | ≫                  |                    |                    |       |   |
| +TM2 <sub>JE</sub> | .113 | ≈    | ≈    | ≈    | <    | <                  | ≫                  | ≈                  |                    |       |   |
| +TM1-              | .132 | ≈    | ≈    | ≈    | ≈    | <                  | >                  | ≈                  | ≈                  |       |   |
| +TM2-              | .074 | ≈    | ≪    | ≪    | ≪    | ≪                  | ≪                  | ≪                  | ≪                  | ≪     | ≪ |
| <b>MAP</b>         |      | .119 | .126 | .128 | .130 | .142               | .104               | .133               | .113               | .132  |   |
|                    |      | +UT  | +TT  | +TM1 | +TM2 | +TM1 <sub>JD</sub> | +TM2 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2 <sub>JE</sub> | +TM1- |   |

Table D.27: Comparison between typed retrieval models on  $Q_F$  ( $\alpha$  tuned using DevSet).

### D.3.3 Post-Hoc Analyses in Section 7.3

| Num.       | Query | SubQ |
|------------|-------|------|------------|-------|------|------------|-------|------|------------|-------|------|
| <b>1</b>   | 8     | 1    | <b>2</b>   | 12    | 1    | <b>3</b>   | 18    | 1    | <b>4</b>   | 19    | 1    |
| <b>5</b>   | 21    | 1    | <b>6</b>   | 25    | 1    | <b>7</b>   | 26    | 2    | <b>8</b>   | 27    | 1    |
| <b>9</b>   | 28    | 2    | <b>10</b>  | 29    | 1    | <b>11</b>  | 31    | 1    | <b>12</b>  | 31    | 2    |
| <b>13</b>  | 31    | 3    | <b>14</b>  | 32    | 2    | <b>15</b>  | 35    | 1    | <b>16</b>  | 36    | 1    |
| <b>17</b>  | 36    | 2    | <b>18</b>  | 39    | 1    | <b>19</b>  | 49    | 1    | <b>20</b>  | 56    | 1    |
| <b>21</b>  | 56    | 2    | <b>22</b>  | 59    | 1    | <b>23</b>  | 62    | 1    | <b>24</b>  | 69    | 1    |
| <b>25</b>  | 72    | 2    | <b>26</b>  | 73    | 2    | <b>27</b>  | 76    | 1    | <b>28</b>  | 76    | 2    |
| <b>29</b>  | 82    | 1    | <b>30</b>  | 82    | 2    | <b>31</b>  | 83    | 1    | <b>32</b>  | 88    | 1    |
| <b>33</b>  | 91    | 1    | <b>34</b>  | 91    | 3    | <b>35</b>  | 92    | 1    | <b>36</b>  | 103   | 2    |
| <b>37</b>  | 108   | 1    | <b>38</b>  | 112   | 1    | <b>39</b>  | 114   | 1    | <b>40</b>  | 117   | 1    |
| <b>41</b>  | 117   | 2    | <b>42</b>  | 120   | 1    | <b>43</b>  | 121   | 1    | <b>44</b>  | 134   | 1    |
| <b>45</b>  | 135   | 1    | <b>46</b>  | 135   | 2    | <b>47</b>  | 135   | 3    | <b>48</b>  | 143   | 1    |
| <b>49</b>  | 148   | 1    | <b>50</b>  | 152   | 1    | <b>51</b>  | 165   | 1    | <b>52</b>  | 167   | 1    |
| <b>53</b>  | 167   | 2    | <b>54</b>  | 175   | 1    | <b>55</b>  | 176   | 2    | <b>56</b>  | 180   | 1    |
| <b>57</b>  | 180   | 2    | <b>58</b>  | 181   | 1    | <b>59</b>  | 187   | 1    | <b>60</b>  | 189   | 1    |
| <b>61</b>  | 189   | 2    | <b>62</b>  | 196   | 1    | <b>63</b>  | 200   | 1    | <b>64</b>  | 205   | 1    |
| <b>65</b>  | 211   | 1    | <b>66</b>  | 218   | 1    | <b>67</b>  | 220   | 1    | <b>68</b>  | 221   | 1    |
| <b>69</b>  | 222   | 1    | <b>70</b>  | 228   | 1    | <b>71</b>  | 228   | 2    | <b>72</b>  | 228   | 3    |
| <b>73</b>  | 231   | 1    | <b>74</b>  | 232   | 1    | <b>75</b>  | 235   | 1    | <b>76</b>  | 237   | 3    |
| <b>77</b>  | 242   | 1    | <b>78</b>  | 246   | 1    | <b>79</b>  | 246   | 2    | <b>80</b>  | 248   | 1    |
| <b>81</b>  | 250   | 1    | <b>82</b>  | 250   | 3    | <b>83</b>  | 253   | 1    | <b>84</b>  | 258   | 3    |
| <b>85</b>  | 264   | 1    | <b>86</b>  | 268   | 1    | <b>87</b>  | 286   | 3    | <b>88</b>  | 287   | 1    |
| <b>89</b>  | 287   | 2    | <b>90</b>  | 301   | 1    | <b>91</b>  | 301   | 2    | <b>92</b>  | 301   | 3    |
| <b>93</b>  | 309   | 1    | <b>94</b>  | 310   | 1    | <b>95</b>  | 311   | 1    | <b>96</b>  | 311   | 2    |
| <b>97</b>  | 317   | 1    | <b>98</b>  | 322   | 1    | <b>99</b>  | 334   | 1    | <b>100</b> | 335   | 1    |
| <b>101</b> | 335   | 2    | <b>102</b> | 348   | 1    | <b>103</b> | 352   | 1    | <b>104</b> | 352   | 2    |
| <b>105</b> | 352   | 3    | <b>106</b> | 354   | 1    |            |       |      |            |       |      |

Table D.28: Mapping of query IDs to CUMTC queries as shown on the x-axis of the difference in reciprocal rank plots in Section 7.3.2.

### D.3.3.1 Post-Hoc Analysis 1

| Subset ID | Subset label                  | Relevant documents | Avg. Formula Density         |              | Avg. Word Density            |              |
|-----------|-------------------------------|--------------------|------------------------------|--------------|------------------------------|--------------|
|           |                               |                    | Q                            | D            | Q                            | D            |
| (0)       | $Q_{FT}$                      | 118                | 10.63                        | 718.53       | 124.17                       | 4669.71      |
| (3)       | $A \cap B$                    | 118                | 10.63                        | 718.53       | 124.17                       | 4669.71      |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 20                 | 9.47                         | 755.0        | 116.16                       | 4621.35      |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 69                 | 12.16                        | 708.36       | 127.23                       | 4406.25      |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 29                 | 7.46                         | 717.59       | 118.04                       | 5329.93      |
| Subset ID | Subset label                  | Relevant documents | Avg. Type Density            |              | Avg. Positive Typing Density |              |
|           |                               |                    | Q                            | D            | Q                            | D            |
| (0)       | $Q_{FT}$                      | 118                | 17.83                        | 879.8        | 2.77                         | 12817505.56  |
| (3)       | $A \cap B$                    | 118                | 17.83                        | 879.8        | 2.77                         | 12817505.56  |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 20                 | 14.53                        | 901.4        | 2.16                         | 11361078.35  |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 69                 | 18.52                        | 801.46       | 3.3                          | 13488774.25  |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 29                 | 17.75                        | 1051.28      | 1.93                         | 12224781.59  |
| Subset ID | Subset label                  | Relevant documents | Avg. Negative typing Density |              | Avg. Typing Edges            |              |
|           |                               |                    | Q                            | D            | Q                            | D            |
| (0)       | $Q_{FT}$                      | 118                | 19.32                        | 135966583.4  | 22.09                        | 148784088.96 |
| (3)       | $A \cap B$                    | 118                | 19.32                        | 135966583.4  | 22.09                        | 148784088.96 |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 20                 | 13.95                        | 120502771.95 | 16.11                        | 131863850.3  |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 69                 | 21.34                        | 143086437.36 | 24.64                        | 156575211.61 |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 29                 | 17.46                        | 129690938.76 | 19.39                        | 141915720.34 |
| Subset ID | Subset label                  | Relevant documents | Avg. total MathML Nodes      |              | Avg. Formula Complexity      |              |
|           |                               |                    | Q                            | D            | Q                            | D            |
| (0)       | $Q_{FT}$                      | 118                | 151.02                       | 17703.15     | 14.89                        | 24.11        |
| (3)       | $A \cap B$                    | 118                | 151.02                       | 17703.15     | 14.89                        | 24.11        |
| (5)       | $A \cap B, rank(A) > rank(B)$ | 20                 | 126.58                       | 15919.4      | 13.17                        | 22.17        |
| (6)       | $A \cap B, rank(B) > rank(A)$ | 69                 | 173.89                       | 19160.23     | 15.00                        | 26.01        |
| (7)       | $A \cap B, rank(B) = rank(A)$ | 29                 | 106.04                       | 15466.48     | 15.07                        | 20.91        |

Table D.29: Subset statistics for investigating the behaviour of Types2XExp+TM1– (model A) and Types2XExp+TM1<sub>JE</sub> (model B) on  $Q_{FT}$  (both tuned using f-t-t).

### D.3.4 Comparison to Textual Models from Section 7.1, DevSet Tuning

|                           | Types2XExp | +TM1-          | +TM2-          | +UT            | +TT            | +TM1           | +TM1 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2           | +TM2 <sub>JD</sub> | +TM2 <sub>JE</sub> |
|---------------------------|------------|----------------|----------------|----------------|----------------|----------------|--------------------|--------------------|----------------|--------------------|--------------------|
| <b>Section 7.1 model</b>  | <b>MAP</b> | .141           | .079           | .109           | .130           | .132           | .148               | .138               | .138           | .111               | .121               |
| VSM                       | .072       | * <b>-.069</b> | -.006          | -.037          | * <b>-.058</b> | * <b>-.060</b> | * <b>-.076</b>     | * <b>-.066</b>     | * <b>-.065</b> | -.039              | -.048              |
| BM25                      | .071       | * <b>-.071</b> | -.008          | -.038          | * <b>-.060</b> | * <b>-.062</b> | * <b>-.078</b>     | * <b>-.068</b>     | * <b>-.067</b> | -.041              | -.050              |
| MLM <sub>jm</sub>         | .088       | * <b>-.054</b> | +0.09          | -.021          | * <b>-.043</b> | * <b>-.045</b> | * <b>-.060</b>     | * <b>-.050</b>     | * <b>-.050</b> | -.024              | -.033              |
| MLM <sub>jm</sub> +RM3    | .066       | * <b>-.075</b> | -.013          | -.043          | * <b>-.065</b> | * <b>-.066</b> | * <b>-.082</b>     | * <b>-.072</b>     | * <b>-.072</b> | -.046              | -.055              |
| MLM <sub>dir</sub>        | .070       | * <b>-.072</b> | -.009          | -.039          | * <b>-.061</b> | * <b>-.063</b> | * <b>-.079</b>     | * <b>-.069</b>     | * <b>-.068</b> | -.042              | -.051              |
| MLM <sub>dir</sub> +RM3   | .081       | * <b>-.060</b> | +0.03          | -.028          | -.049          | -.051          | * <b>-.067</b>     | * <b>-.057</b>     | * <b>-.056</b> | -.030              | -.040              |
| MLM <sub>dir</sub> +PRM2  | .075       | * <b>-.066</b> | -.004          | -.034          | -.055          | -.057          | * <b>-.073</b>     | * <b>-.063</b>     | * <b>-.062</b> | -.036              | -.046              |
| MLM <sub>dir</sub> '      | .070       | * <b>-.071</b> | -.009          | -.039          | * <b>-.061</b> | * <b>-.063</b> | * <b>-.078</b>     | * <b>-.068</b>     | * <b>-.068</b> | -.042              | * <b>-.051</b>     |
| MLM <sub>dir</sub> ' +RM3 | .036       | * <b>-.105</b> | * <b>-.043</b> | * <b>-.073</b> | * <b>-.094</b> | * <b>-.096</b> | * <b>-.112</b>     | * <b>-.102</b>     | * <b>-.101</b> | * <b>-.075</b>     | * <b>-.085</b>     |
| SPUD                      | .095       | -.046          | +0.16          | -.014          | -.036          | -.038          | * <b>-.053</b>     | -.043              | -.043          | -.017              | -.026              |
| SPUD+RM3                  | .035       | * <b>-.106</b> | * <b>-.044</b> | * <b>-.074</b> | * <b>-.095</b> | * <b>-.097</b> | * <b>-.113</b>     | * <b>-.103</b>     | * <b>-.102</b> | * <b>-.076</b>     | * <b>-.086</b>     |
| SPUD+PRM2                 | .070       | * <b>-.071</b> | -.009          | -.039          | * <b>-.060</b> | * <b>-.062</b> | * <b>-.078</b>     | * <b>-.068</b>     | * <b>-.067</b> | -.051              | -.051              |
| TF-IDFExp                 | .047       | * <b>-.094</b> | -.032          | * <b>-.062</b> | * <b>-.083</b> | * <b>-.085</b> | * <b>-.101</b>     | * <b>-.091</b>     | * <b>-.090</b> | * <b>-.064</b>     | * <b>-.074</b>     |
| Types2X                   | .073       | * <b>-.068</b> | -.006          | -.036          | -.057          | -.059          | * <b>-.075</b>     | * <b>-.065</b>     | -.065          | -.038              | -.048              |
| Types2XExp                | .154       | +0.13          | * <b>+.076</b> | +0.45          | +0.24          | +0.22          | +0.06              | +0.16              | +0.17          | * <b>+.043</b>     | * <b>+.033</b>     |

Table D.30: Comparison of typed retrieval models (parameters tuned on the DevSet) with the textual retrieval models from Section 7.1 on  $Q_{FT}$  (106 queries).

|                           | Types2XExp | +TM1-          | +TM2-          | +UT            | +TT            | +TM1           | +TM1 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2           | +TM2 <sub>JD</sub> | +TM2 <sub>JE</sub> |
|---------------------------|------------|----------------|----------------|----------------|----------------|----------------|--------------------|--------------------|----------------|--------------------|--------------------|
| <b>Section 7.1 model</b>  | <b>MAP</b> | .132           | .074           | .119           | .126           | .128           | .142               | .133               | .130           | .104               | .113               |
| VSM                       | .069       | * <b>-.063</b> | -.005          | -.050          | * <b>-.057</b> | * <b>-.059</b> | * <b>-.073</b>     | * <b>-.064</b>     | * <b>-.061</b> | -.035              | -.044              |
| BM25                      | .072       | * <b>-.060</b> | -.002          | -.047          | -.054          | * <b>-.056</b> | * <b>-.070</b>     | * <b>-.061</b>     | * <b>-.058</b> | -.032              | -.040              |
| MLM <sub>jm</sub>         | .091       | -.041          | +0.17          | -.028          | -.035          | -.037          | * <b>-.051</b>     | -.042              | -.039          | -.013              | -.022              |
| MLM <sub>jm</sub> +RM3    | .074       | -.058          | -.000          | -.045          | -.053          | * <b>-.054</b> | * <b>-.068</b>     | * <b>-.059</b>     | * <b>-.056</b> | -.030              | -.039              |
| MLM <sub>dir</sub>        | .076       | * <b>-.057</b> | +0.02          | -.044          | * <b>-.051</b> | * <b>-.052</b> | * <b>-.066</b>     | * <b>-.058</b>     | * <b>-.054</b> | -.029              | -.037              |
| MLM <sub>dir</sub> +RM3   | .090       | -.042          | +0.16          | -.029          | -.036          | -.038          | -.052              | -.043              | -.040          | -.014              | -.023              |
| MLM <sub>dir</sub> +PRM2  | .078       | -.054          | +0.04          | -.041          | -.048          | -.050          | * <b>-.064</b>     | * <b>-.055</b>     | -.052          | -.026              | -.035              |
| MLM <sub>dir</sub> '      | .075       | * <b>-.058</b> | +0.01          | -.045          | * <b>-.052</b> | * <b>-.053</b> | * <b>-.067</b>     | * <b>-.059</b>     | * <b>-.055</b> | -.030              | -.038              |
| MLM <sub>dir</sub> ' +RM3 | .046       | * <b>-.086</b> | -.028          | * <b>-.073</b> | * <b>-.080</b> | * <b>-.082</b> | * <b>-.096</b>     | * <b>-.087</b>     | * <b>-.084</b> | * <b>-.058</b>     | * <b>-.067</b>     |
| SPUD                      | .098       | -.034          | +0.24          | -.021          | -.028          | -.030          | -.044              | -.035              | -.032          | -.006              | -.014              |
| SPUD+RM3                  | .044       | * <b>-.088</b> | -.030          | * <b>-.075</b> | * <b>-.082</b> | * <b>-.084</b> | * <b>-.098</b>     | * <b>-.089</b>     | * <b>-.086</b> | * <b>-.060</b>     | * <b>-.069</b>     |
| SPUD+PRM2                 | .073       | * <b>-.059</b> | -.001          | -.046          | * <b>-.053</b> | * <b>-.055</b> | * <b>-.069</b>     | * <b>-.060</b>     | * <b>-.057</b> | -.031              | -.040              |
| TF-IDFExp                 | .053       | * <b>-.079</b> | -.021          | * <b>-.066</b> | * <b>-.073</b> | * <b>-.074</b> | * <b>-.089</b>     | * <b>-.080</b>     | * <b>-.076</b> | * <b>-.051</b>     | * <b>-.059</b>     |
| Types2X                   | .071       | * <b>-.061</b> | -.003          | -.048          | -.055          | -.057          | * <b>-.071</b>     | * <b>-.062</b>     | * <b>-.059</b> | -.033              | -.042              |
| Types2XExp                | .156       | +0.24          | * <b>+.082</b> | +0.37          | * <b>+.029</b> | +0.28          | +0.14              | +0.23              | * <b>+.026</b> | * <b>+.052</b>     | * <b>+.043</b>     |

Table D.31: Comparison of typed retrieval models (parameters tuned on DevSet) with the textual retrieval models from Section 7.1 on  $Q_F$  (116 queries).

|                           | Types2XExp | +TM1-   | +TM2-   | +UT     | +TT     | +TM1    | +TM1 <sub>JD</sub> | +TM1 <sub>JE</sub> | +TM2    | +TM2 <sub>JD</sub> | +TM2 <sub>JE</sub> |
|---------------------------|------------|---------|---------|---------|---------|---------|--------------------|--------------------|---------|--------------------|--------------------|
| <b>Section 7.1 model</b>  | <b>MAP</b> | .163    | .120    | .153    | .158    | .160    | .170               | .163               | .161    | .142               | .149               |
| VSM                       | .076       | **-.087 | -.044   | **-.077 | **-.082 | **-.084 | **-.094            | **-.087            | **-.085 | **-.066            | **-.073            |
| BM25                      | .077       | **-.085 | -.043   | **-.076 | **-.081 | **-.082 | **-.093            | **-.086            | **-.084 | **-.065            | **-.071            |
| MLM <sub>jm</sub>         | .084       | **-.078 | -.036   | **-.069 | **-.074 | **-.075 | **-.085            | **-.079            | **-.077 | **-.058            | **-.064            |
| MLM <sub>jm</sub> +RM3    | .063       | **-.100 | *-.058  | **-.090 | **-.096 | **-.097 | **-.107            | **-.101            | **-.098 | **-.080            | **-.086            |
| MLM <sub>dir</sub>        | .066       | **-.097 | *-.054  | **-.087 | **-.092 | **-.094 | **-.104            | **-.095            | **-.095 | **-.076            | **-.083            |
| MLM <sub>dir</sub> +RM3   | .082       | **-.081 | -.038   | *-.071  | **-.076 | **-.078 | **-.088            | **-.081            | **-.079 | *-.060             | *-.067             |
| MLM <sub>dir</sub> +PRM2  | .061       | **-.101 | *-.059  | **-.092 | **-.097 | **-.098 | **-.108            | **-.102            | **-.100 | **-.081            | **-.087            |
| MLM <sub>dir</sub> '      | .072       | **-.090 | *-.048  | **-.081 | **-.086 | **-.087 | **-.097            | **-.091            | **-.089 | **-.070            | **-.076            |
| MLM <sub>dir</sub> ' +RM3 | .051       | **-.112 | **-.070 | **-.102 | **-.107 | **-.109 | **-.119            | **-.112            | **-.110 | **-.091            | **-.098            |
| SPUD                      | .090       | **-.072 | -.030   | *-.063  | **-.068 | **-.069 | **-.079            | **-.073            | **-.070 | *-.052             | *-.058             |
| SPUD+RM3                  | .050       | **-.113 | **-.070 | **-.103 | **-.108 | **-.109 | **-.120            | **-.113            | **-.111 | **-.092            | **-.098            |
| SPUD+PRM2                 | .072       | **-.091 | *-.049  | **-.082 | **-.087 | **-.088 | **-.098            | **-.092            | **-.089 | **-.071            | **-.077            |
| TF-IDFExp                 | .060       | **-.102 | *-.060  | **-.093 | **-.098 | **-.099 | **-.109            | **-.103            | **-.101 | **-.082            | **-.088            |
| Types2X                   | .070       | **-.093 | *-.051  | **-.083 | **-.089 | **-.090 | **-.100            | **-.094            | **-.091 | **-.073            | **-.079            |
| Types2XExp                | .173       | +.010   | **+.053 | +.020   | +.015   | +.013   | +.003              | +.010              | +.012   | **+.031            | +.024              |

Table D.32: Comparison of typed retrieval models (parameters tuned on the DevSet) with the textual retrieval models from Section 7.1 on  $Q$  (160 queries).

### D.3.5 Summary Model Bar plots

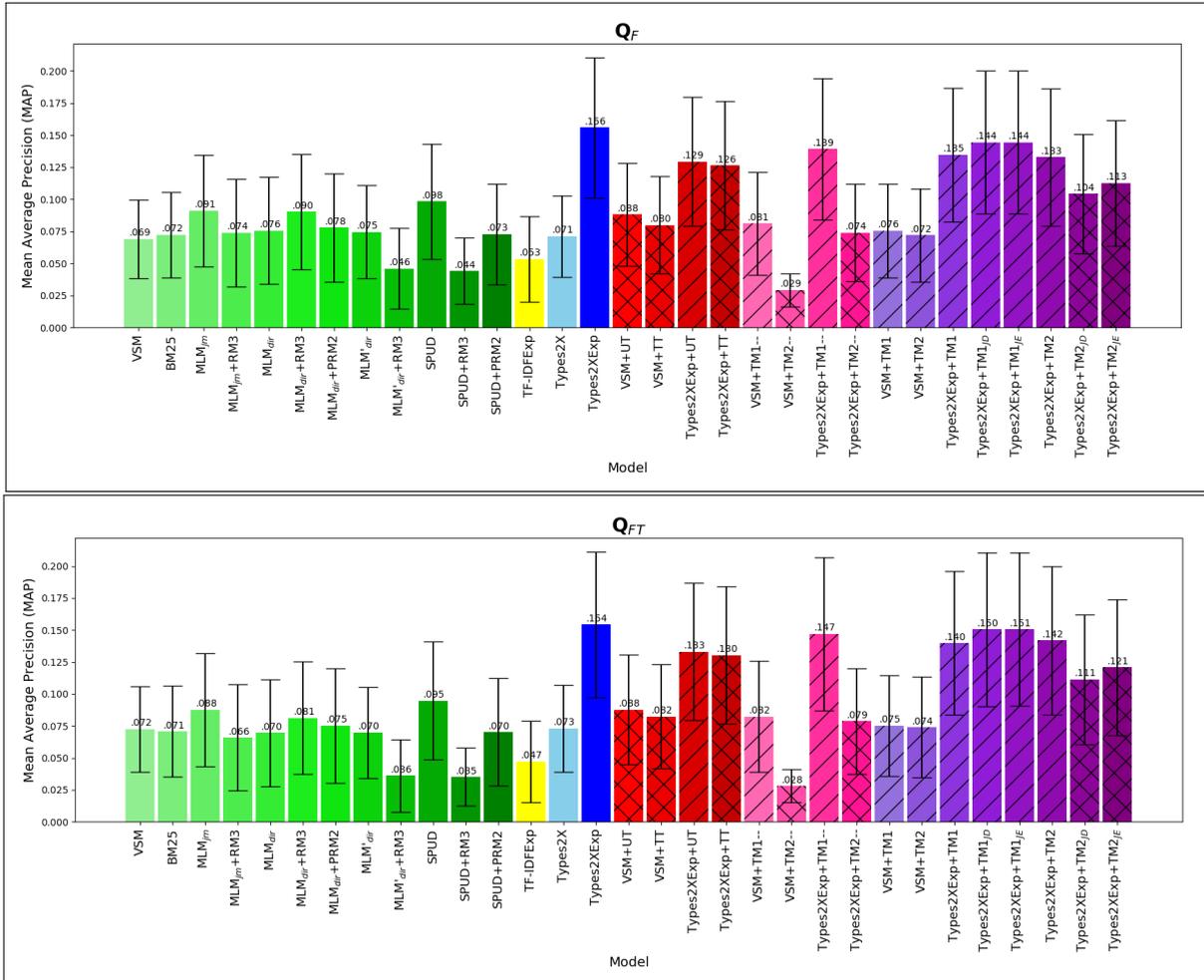


Figure D.1: Summary of models' retrieval results on  $Q_F$  and  $Q_{FT}$ , in MAP.