Variable Typing: Assigning Meaning to Variables in Mathematical Text

Yiannos A. Stathopoulos[♠] **Simon Baker**[♠♣] **Marek Rei**[♠♢] **Simone Teufel**[♠]
[♠]Computer Laboratory, University of Cambridge, United Kingdom
[♣]Language Technology Lab, University of Cambridge, United Kingdom
[♢]The ALTA Institute, University of Cambridge, United Kingdom
{yiannos.stathopoulos,simon.baker,marek.rei,simone.teufel}@cl.cam.ac.uk

Abstract

Information about the meaning of mathematical variables in text is useful in NLP/IR tasks such as symbol disambiguation, topic modeling and mathematical information retrieval (MIR). We introduce *variable typing*, the task of assigning one *mathematical type* (multi-word technical terms referring to mathematical concepts) to each variable in a sentence of mathematical text. As part of this work, we also introduce a new annotated data set composed of 33,524 data points extracted from scientific documents published on arXiv. Our intrinsic evaluation demonstrates that our data set is sufficient to successfully train and evaluate current classifiers from three different model architectures. The best performing model is evaluated on an extrinsic task: MIR, by producing a *typed formula index*. Our results show that the best performing MIR models make use of our typed index, compared to a formula index only containing raw symbols, thereby demonstrating the usefulness of variable typing.

1 Introduction

Scientific documents, such as those from Physics and Computer Science, rely on mathematics to communicate ideas and results. Written mathematics, unlike general text, follows strong domain-specific conventions governing how content is presented. According to Ganesalingam (2008), the sense of mathematical text is conveyed through the interaction of two contexts: the textual context (flowing text) and the mathematical (or symbolic) context (mathematical formulae).

In this work, we introduce a new task that focuses on one particular interaction: the assignment of meaning to variables by surrounding text in the same sentence[1]. For example, in the sentence

> *Let P be a parabolic subgroup of $GL(n)$ with Levi decomposition $P = MN$, where N is the unipotent radical.*

the variables P and N in the symbolic context are assigned the meaning "parabolic subgroup" and "unipotent radical" by the textual context surrounding them respectively.

We will refer to the task of assigning one *mathematical type* to each variable in a sentence as *variable typing*. We use mathematical types (Stathopoulos and Teufel, 2016) as variable denotation labels. Types are multi-word phrases drawn from the technical terminology of the mathematical discourse that label mathematical objects (e.g., "set"), algebraic structures (e.g., "monoid") and instantiable notions (e.g., "cardinality of a set"). In the sentence presented earlier, the phrases "parabolic subgroup", "Levi decomposition" and "unipotent radical" are examples of types.

Typing variables may be beneficial to other natural language processing (NLP) tasks, such as topic modeling, to group documents that assign meaning to variables consistently (e.g., "E" is "energy" consistently in some branches of Physics). In mathematical information retrieval (MIR), for instance, enriching formulae with types may improve precision. For example, the formulae $x + y$ and $a + b$ can be considered α-equivalent matches. However, if a and b are matrices while x and y are vectors, the match is likely to be a false positive. Typing information may be helpful in reducing such instances and improving retrieval precision.

Variable typing differs from similar tasks in three fundamental ways. First, meaning – in the form of mathematical types – is explicitly assigned to variables, rather than arbitrary mathematical expressions. Second, variable typing is carried out at the sentential level, with valid type assignments for variables drawn from the sentences in which

[1]Data for the task is available at https://www.cst.cam.ac.uk/~yas23/

they occur, rather than from larger contexts, such as documents. Third, denotations are drawn from a pre-determined list of types, rather than from free-form text in the surrounding context of each variable.

As part of our work, we have constructed a new data set for variable typing that is suitable for machine learning (Section 4) and is distributed under the Open Data Commons license. We propose and evaluate three models for typing variables in mathematical documents based on current machine learning architectures (Section 5). Our intrinsic evaluation (Section 6) suggests that our models significantly outperform the state-of-the-art SVM model by Kristianto et al. (2012, 2014) (originally developed for description extraction) on our data set. More importantly, our intrinsic evaluation demonstrates that our data set is sufficient to successfully train and evaluate classifiers from three different architectures. We also demonstrate that our variable typing task and data are useful in MIR in our extrinsic evaluation (Section 7).

2 Related Work

The task of extracting semantics for variables from the linguistic context was first proposed by Grigore et al. (2009) with the intention of disambiguating symbols in mathematical expressions. Grigore et al. 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/subscripted 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%.

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). However, Kristianto et al. (2012) note that domain-agnostic parsers are confused by mathematical expressions making rule-based methods sensitive to parse tree errors. Both rule-based extraction methods were outperformed by Support Vector Machines (SVMs) (Kristianto et al., 2012, 2014).

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. (2016) take a ranked approach at 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.

In contrast, we assign meaning exclusively to variables, using denotations from a pre-computed dictionary of mathematical types, rather than free-form text. Types as pre-identified, compositionally constructed denotational labels enable efficient determination of relatedness between mathematical concepts. In our extrinsic MIR experiment (Section 7), the mathematical concept that two or more types are derived from is identified by locating their common parent type – the supertype – on a suffix trie. Topically related types that do not share a common supertype can be identified using an automatically constructed type embedding space (Stathopoulos and Teufel (2016), Section 5.1), rather than manually curated namespaces or fuzzy term clusters.

3 The Variable Typing Task

We define the task of variable typing as follows. Given a sentence containing a pre-identified set of

variables V and types T, variable typing is the task of classifying all edges $V \times T$ as either existent (positive) or non-existent (negative).

However, not all elements of $V \times T$ are valid edges. Invalid edges are usually instances of *type parameterisation*, where some type is parameterised by what appears to be a variable. For example, the set of candidate edges for the sentence

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

would include $(\boldsymbol{V}, \text{exterior algebra})$ and $(\boldsymbol{V^*}, \text{exterior algebra})$ but *not* $(\boldsymbol{q}, \text{exterior algebra})$. Such edges are identified using pattern matching (Java regular expressions) and are not presented to annotators or recorded in the data set.

Our definition of "variable" mirrors that of "simple expression" proposed by Grigore et al. (2009): instances of formulae in the discourse are considered to be "typeable variables" if they are only composed of a single, potentially scripted base identifier.

Variable typing, as defined in this work, is based on four assumptions: (1) typings occur at the sentential level and variables in a sentence can only be assigned a type phrase occurring in that sentence, (2) variables and types in the sentence are known a priori, (3) edges in each sentence are independent of one another, and (4) edges in one sentence are independent of those 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 decision to constrain variable typing at the sentential level is motivated by empirical studies (Grigore et al., 2009; Gödert, 2012). Grigore et al. (2009) have shown that the majority of variables are introduced and declared in the same sentence. In addition, mathematical text tends to be composed of local contexts, such as theorems, lemmas and proofs (Ganesalingam, 2008).

The assumptions introduced above simplify the task of variable typing without sacrificing the generalisability of the task. For example, cases where the same variable is assigned multiple conflicting types from different sentences within a document can be collected and resolved using a *type disambiguation* algorithm.

4 Variable Typing Data Set

We have constructed an annotated data set of sentences for building variable typing classifiers. The sentences in our corpus are sourced from the Mathematical REtrieval Corpus (MREC) (Líška et al., 2011), a subset of arXiv (over 439,000 papers) with all LaTeX formulae converted to MathML.

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

Table 1: Data set statistics.

The data set is split into a standard training/development/test machine learning partitioning scheme as outlined in Table 1. The idea behind this scheme is to train and evaluate new models on standardised data partitions so that results can be directly comparable.

4.1 Sentence Sampling

The structure and role of sentences in mathematical papers may vary according to their location in the discourse. For example, sentences in the "Introduction" – intended to introduce the subject matter – can be expected to differ in structure from those in a proof, which tend to be short, formal statements. Our sampling strategy is designed to control for this diversity in sentence structure. First, we sentence-tokenised and transformed each document in the MREC into a graph that encodes its section structure. Document graphs also take into account blocks of text unique to the mathematical discourse such as theorems, proofs and definitions. Then, we sampled sentences for our data set by distribution according to their location in the source arXiv document.

Variables in each MREC document are identified via a parser that recognises the variable description given in Section 3. Our variable parser is designed to operate on *symbol layout trees* (SLTs) (Schellenberg et al., 2012) – trees representing the 2-dimensional presentation layout of mathematical formulae. We identified 28.6 million sentences that contain variables.

The distribution of sentences according to (a) the type of discourse/math block of origin and (b) the number of unique types in the sentence is reconstructed by putting sentences into bins based

on the value of these features. Sentences are selected from the bins at random in proportion to their size. The training, development and test samples have been produced via repeated application of this sample-by-distribution strategy over the set of all sentences that contain variables.

4.2 Extended Type Dictionary

The type dictionary distributed by Stathopoulos and Teufel (2016) contains 10,601 automatically detected types from the MREC. However, the MREC contains 2.9 million distinct technical terms, many of which might also be types. Therefore, the seed dictionary is too small to be used with variable typing at scale since types from the seed dictionary will be sparsely present in sampled sentences. To overcome this problem, we used the *double suffix trie algorithm* (DSTA) to automatically expand the type dictionary. The algorithm makes use of the fact that most types are compositional (Stathopoulos and Teufel, 2016): longer subtypes can be constructed out of shorter supertypes by attaching pre-modifiers (e.g., a "Riemannian manifold" can be considered a subtype of "manifold").

The DSTA takes two lists of technical terms as input – the seed dictionary of types and the MREC master list (2.9 million technical terms). First, technical terms on both lists are word-tokenised. Then, all technical terms in the seed dictionary (the known types) are placed onto the *known types suffix trie* (KTST). Additional types are generated from single word types on the KTST by expanding them with one of 40 prefixes observed in the corpus. For example, the type "algebra" might generate the supertype "coalgebra". These are also added on the KTST as known types.

Technical terms in the KTST are copied onto the *candidate type suffix trie* (CTST) and are labeled as types. Next, the technical terms on the master list are inserted into the CTST. Technical terms in the master list that have known types from the seed dictionary as their suffix on the CTST are also marked as types. A new dictionary of types (in the form of a list of technical terms) is produced by traversing the CTST and recording all phrases that have a known type as their suffix. This way, we have expanded the type dictionary from 10,601 types to approximately 1.23 million technical terms, from which an updated KTST can be produced.

4.3 Human Annotation and Agreement

Two of the authors jointly developed the annotation scheme and guidelines using sentences sampled by distribution as discussed in Section 4.1. Sentences sampled for this purpose are excluded from subsequent sampling. The labeling scheme, presented in Table 2, implements the assumptions of the variable typing task – each variable in a sentence is assigned exactly one label: either one type from the sentence or one of six fixed labels for special situations.

An annotation experiment was carried out using two authors as annotators to investigate (a) how intuitive the task of typing is to humans and (b) the reliability of the annotation scheme. For this purpose, a further 1,000 sentences were sampled (and removed) from the pool and organised into two subsamples each with 554 sentences. The subsamples have an overlap of 108 sentences with a total of 182 edges, which are used to measure inter-annotator agreement.

We report annotator agreement for three separate cases. The first case reflects whether 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 2. In this case, agreement is substantial (Cohen's $K = 0.80$, $N = 182$, $k = 2$, $n = 2$). 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, we report accuracy as the proportion of decisions where annotators agree over all decisions: 90.9%. In the last case where both annotators agree that a variable is not a type (i.e., is assigned one of the six fixed labels), agreement has been found to be moderate (Fleiss' $K = 0.61$, $N = 123$, $k = 2$, $n = 6$).

The bulk of the annotation was carried out by one of the author-annotators and was produced by repeated sampling by distribution (as described in Section 4.1). Sentences in the bulk sample are combined with the 554 sentences annotated by the author during the annotation experiment to produce a final data set composed of 7,803 sentences. The training, test and development sets have been produced using the established 70% for training,

Label	Description
One label per type instance	One label per instance of any type 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 2: Labels for special typing situations.

20% for test and 10% for development data set partitioning strategy. Each partition is sampled by distribution in order to model training and predicting typings over complete discourse units, such as documents.

5 Experiments

We compare three models for variable typing to two baselines: the "nearest type" baseline and the SVM proposed by Kristianto et al. (2014). One of our models is an extension of the latter baseline with both type and variable-centric features. The other two models are based on deep neural networks: a convolutional neural network and a bidirectional LSTM.

We treat the task of typing as binary classification: every possible typing in a sentence is presented to a classifier which, in turn, is expected to make a "type" or "not-type" decision. We say that an edge is *positive* if it connects a variable to a type in the sentence and *negative* otherwise.

5.1 Computing a Type Embedding Space

We use the extended dictionary of types (Section 4.2) to pre-train a *type embedding space*. Computed over the MREC, a type embedding space includes embeddings for both words and types (as atomic lexical tokens). These vectors are used by our deep neural networks to model the distributional meaning of words and types. The type embedding space is constructed using the process described by Stathopoulos and Teufel (2016): occurrences of extended dictionary type phases in the MREC are substituted with unique atomic lexical units before the text is passed on to `word2vec`.

5.2 Models for Variable Typing

Nearest Type baseline (NT) 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. This baseline is intended to approximate the "nearest noun" baseline (Kristianto et al., 2012, 2014) which we cannot directly compute due to the fact that noun phrases in the text become parts of types.

Support Vector Machine (Kristianto et al.) (SVM) This is an implementation of the features and linear SVM described by Kristianto et al. (2012). Furthermore, we use the same value for hyperparameter C (the soft margin cost parameter) used by Kristianto et al. (2012). Due to the class imbalance in our data set we have used inversely proportional class weighting (as implemented in scikit-learn). L2-normalisation is also applied.

Extended Support Vector Machine (SVM+) We have extended the SVM proposed by Kristianto et al. (2012) with the features that are type and variable-centric, such as the 'base symbol of a candidate variable' and 'first letter in the candidate type'. A description of these extended features are listed in Table 4. We applied automatic class weighting and L2-normalisation. We have found that $C = 2$ is optimal for this model by fine-tuning over the development set.

Convolutional Neural Network (Convnet) We use 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 the SVM models, we do not use any hand-crafted features, but only the inputs (Table 3), and the pre-trained embeddings (Section 5.1).

The input is a tensor that encodes the input described in Table 3. We use the embeddings to represent the input tokens. In addition, we concatenate two dimensions to the input for each token : one dimension to denote (using 1 or 0) whether a given token is a type 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

Name	Description
Token	A word in the sentence. If the token is a formula (including a variable), the token is '@@@'. Types are represented by the key of their embedding vector.
Token class	An integer – 0 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	If the token is a type and it is part of the edge being considered, this field takes the value 'TYPE' or '-' otherwise.

Table 3: Input and features to neural network typing models.

Orientation	Description
Type	Number of words in the candidate type.
Type	The base type of each candidate type.
Type and Variable	The first letter in the type and base symbol of the candidate variable.
Type	The grammatical number of the type as it appears in the sentence.
Variable	The variables and symbols in the candidate variable layout graph (one string per symbol).
Variable	The number of distinct symbols in the candidate variable layout graph.
Variable	The base symbol of the candidate variable layout graph.
Variable	The directions (Above, Below,Up-left, Up-right, Down-left, Down-right, Next) in which a candidate symbol has neigbouring symbols.
Variable	Operators in the mathematical context of the candidate variable layout graph.
Sentence	Prefix sequence: tokens from start of sentence to a (exclusive)
Sentence	Middle sequence: tokens between a and b (exclusive)
Sentence	Suffix sequence: tokens between b (exclusive) and end of sentence.

Table 4: 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).

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 using binary cross-entropy as loss function, and the ADAM optimizer (Kingma and Ba, 2014). We tune the aforementioned hyperparameters on the development data and we use balanced oversampling with replacement in order to adjust for the class imbalance in the data.

Our 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 takes as input a sequence of words, which are then mapped to word embeddings. For each token in the input sentence, we also include the inputs described in Table 3. In addition, the model uses one string feature we refer to as "supertype". If the token is a type, then this feature is the string key of the embedding vector of its supertype or "NONE" otherwise.

These features are mapped to a separate embedding space and then concatenated with the word embedding to form a single task-specific word representation. This allows us to capture useful information about each word, and also designate which words to focus on when processing the sentence.

We use a neural sequence labeling architecture, 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, 1997), and a context-specific representation of each word is created by concatenating the hidden representations from both directions.

A hidden layer is added on top to combine the features from both directions. Finally, we use a softmax output layer that predicts a probability distribution over positive or negative assignment for a given edge.

We also make use of an extension of neural sequence labeling that combines character-based word representations with word embeddings using a predictive gating operation (Rei et al., 2016). This allows our model to capture character-level patterns and estimate representations for previously unseen words.

In this framework, an alternative word repre-

sentation 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 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 and 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.

6 Intrinsic Evaluation

Evaluation is performed over edges, rather than sentences, in the test set. We measure performance using precision, recall and F_1-score. We use the non-parametric *paired randomisation test* to detect significant differences in performance across classifiers.

The convnet and BiLSTM models are trained and evaluated with as many sentences as there are edges: the source sentence is copied for each input edge, with inputs modified to reflect the relation of interest. We employed early stopping and dropout to avoid overfitting with these models.

Table 5 shows the performance results of all classifiers considered. All three proposed models have significantly outperformed the NT baseline and Kristianto et al.'s (Kristianto et al., 2014) state-of-the-art SVM. The best performing model is the bidirectional LSTM ($F_1 = 78.98\%$) which has significantly outperformed all other models ($\alpha = 0.01$).

According to the results in Table 5, 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.

The neural models outperform SVM+ despite the fact that the latter is a product of laborious manual feature engineering. In contrast, no manual feature engineering has been performed on the Convnet model (or indeed on any of the deep neural network models).

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

Table 5: Model performance summary. All figures are statistically significant ($p < 0.01$) according to the randomisation test.

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.

7 Extrinsic Evaluation

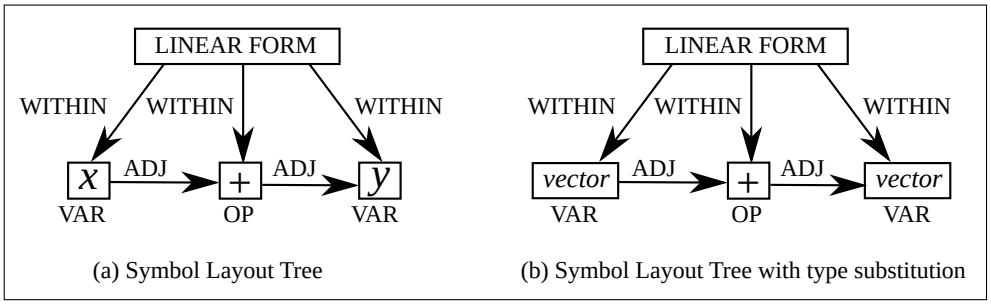
We demonstrate that our data set and variable typing task are useful using a mathematical information retrieval (MIR) experiment. The hypothesis for our MIR experiment is two-fold: (a) types identified in the textual context for the variable typing task are also useful for text-based mathematical retrieval and (b) substituting raw symbols with types in mathematical expressions will have an observable effect to MIR.

In order to motivate the second hypothesis, consider the following natural language query:

> *Let x be a vector. Is there another vector y such that $x + y$ will produce the zero element?*

In the context of MIR, mathematical expressions are represented using SLTs (Pattaniyil and Zanibbi, 2014) that are constructed by parsing presentation MathML. The expression "$x + y$" is represented by the SLT in figure 1(a). The variable typing classifier and the type disambiguation algorithm determine the types of the variables x and y as "vector". Thus, the variable nodes in figure 1(a) will be substituted with their type, producing the SLT in figure 1(b).

The example query can be satisfied by identifying a vector y such that when added to x will produce the zero vector. This operation is abstract in mathematics and extends to objects beyond vectors, including integers. In an untyped formula index, there is no distinction between instances of $x + y$ where the variables are integers or vectors. As a result, documents where both variables are integers might also be returned. In contrast,



(a) Symbol Layout Tree (b) Symbol Layout Tree with type substitution

Figure 1: (a) SLT representation of the expression $x + y$, (b) typed SLT for the expression $x + y$.

a typed formula index will return instances of the typed SLT in figure 1(b) where the variables are vectors, as opposed to integers. Therefore, a typed index can reduce the number of false positives and increase precision.

Four MIR retrieval models are introduced in Section 7.3 designed to control for text indexing/retrieval so that the effects of type-aware vs type-agnostic formula indexing and scoring can be isolated. These models make use of the Tangent formula indexing and scoring functions (Pattaniyil and Zanibbi, 2014), which we have implemented.

We use the Cambridge University Math IR Test Collection (CUMTC) (Stathopoulos and Teufel, 2015) which is composed of 120 research-level mathematical information needs and 160 queries. The CUMTC is ideal for our evaluation for two reasons. First, topics in the CUMTC are expressed in natural language and are rich in mathematical types. This allows us to directly apply our best performing variable typing model (BiLSTM) in our retrieval experiment in order to extract variable typings for documents and queries. Second, the CUMTC uses the MREC as its underlying document collection, which enables downstream evaluation in an optimal setting for variable typing.

7.1 Tangent Formula Indexing and Scoring

Given a mathematical formula, the Tangent indexing algorithm starts from the root node of an SLT and generates symbol pair tuples in a depth-first manner. 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 traversal, the index is updated to record one tuple representing the relationship between the current node and every node in the path to the SLT root. We have also implemented Tangent's method of indexing matrices, but we refer the reader to Pattaniyil and Zanibbi (2014) for further details.

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) 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 $e_1, \ldots, e_n$ is

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

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)):

$$\lambda \times L(d) + (1 - \lambda) \times m(d, e_1, \ldots, e_n)$$

7.2 Typed Tangent Indexing and Scoring

We have applied the BiLSTM variable typing model to obtain variable typings for all symbols in the documents in the MREC. For each document in the collection our adapted Tangent formula indexer first groups the variable typing edges for that document according to the variable identifier involved. Subsequently, our typed indexing process applies a type disambiguation algorithm to determine which of the candidate types associated with the variable will be designated as its type.

For a variable v in document d, our type disambiguation algorithm first looks at the known types suffix trie (KTST) containing all 1.23 million types in order to find a common parent be-

tween the candidate types. If a common supertype T is discovered, then v is said to be of type T. Otherwise, the type disambiguation algorithm uses simple majority vote amongst the candidates to determine the final type for variable v.

The type disambiguation algorithm is applied to every typing group until all variable typings have been processed. Variable groups with no type candidates (e.g., no variable typings have been extracted for a variable) are assigned a missing type symbol ("*"). Subsequently, variables in the SLT of each formula in d are replaced with their type or the missing type symbol. An index, referred to as the typed index, is generated by applying the tangent indexing process on the modified SLTs.

The same process is applied to query formulae during query time in order to facilitate typed matching and scoring.

7.3 Results

We have replicated runs of the Lucene vector-space model (VSM) and BM25 models presented by Stathopoulos and Teufel (2016) on the CUMTC. Furthermore, we introduce four models based on Tangent indexing and scoring that represent different strategies in handling types in text and formulae. We refer to a model as *typed* if it uses the type-substituted version of the Tangent index and *untyped* otherwise.

Text with types removed (RT): The Lucene score $L(d)$ is computed over a text index with type phrases completely removed. This model is intended to isolate the performance of retrieval on the formula index alone. We consider both typed and untyped instances of this model.

Text with types(TY): The Lucene score is computed over a text index that treats type phrases as atomic lexical tokens. This model is intended to simulate type-aware text that enables the application of variable typing. Both typed and untyped instances of this model are considered.

Optimal values for the linear combination parameter λ are obtained using 13 queries in the "development set" of the CUMTC. We report mean average precision (MAP) for our models computed over all 160 queries in the main CUMTC. MAPs obtained over the CUMTC are low due to the difficulty of the queries rather than an unstable evaluation (Stathopoulos and Teufel, 2016). The paired randomisation test is used to test for significance in retrieval performance gains between the models.

	VSM	**BM25**		
MAP	.076	.079		
	RT typed	**RT untyped**	**TY typed**	**TY untyped**
MAP	.046	.052	.139	.083
λ_{opt}	.9	.9	.9	.4

Table 6: MIR model performance summary.

The results of our MIR experiments are presented in Table 6. The best performing model is TY/typed which significantly outperforms all other baselines ($p - value < 0.05$ for comparison with BM25 and $p - value < 0.01$ with all other models). The TY/typed model yields almost double the MAP performance of its untyped counterpart (TY/untyped, .083 MAP). In contrast, the RT/typed and RT/untyped models perform comparably (no significant difference) but poorly. This drop in MAP performance suggests that type phrases are beneficial for text-based retrieval of mathematics. Retrieval models employing formula indexing seem to be affected by both the presence of types in the text as well as in the formula index. The TY/typed model outperforms the TY/untyped model, which in turn outperforms RT/untyped. This suggests that gains in retrieval performance are strongest when types are used in both text and formula retrieval – models using either approach alone do not perform as well. These results demonstrate that variable typing is a valuable task in MIR.

8 Conclusions

This work introduces the new task of variable typing and an associated data set containing 33,524 labeled edges in 7,803 sentences. We have constructed three variable typing models and have shown that they outperform the current state-of-the-art methods developed for similar tasks. The BiLSTM model is the top performing model achieving 79% F_1-score. This model is then evaluated in an extrinsic downstream task–MIR, where we augmented Tangent formula indexing with variable typing. A retrieval model employing the typed Tangent index outperforms all considered retrieval models demonstrating that our variable typing task, data and trained model are useful in downstream applications. We make our variable typing data set available through the Open Data Commons license.

References

Mohan Ganesalingam. 2008. *The Language of Mathematics*. Ph.D. thesis, Cambridge University Computer Laboratory.

Winfried Gödert. 2012. Detecting multiword phrases in mathematical text corpora. *CoRR*, abs/1210.0852.

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.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural Computation*, 9.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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, Goran Topic, and Akiko Aizawa. 2014. Exploiting textual descriptions and dependency graph for searching mathematical expressions in scientific papers.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of NAACL-HLT 2016*.

Martin Líška, Petr Sojka, Michal Růžička, and Petr Mravec. 2011. Web interface and collection for mathematical retrieval: Webmias and mrec. In *Towards a Digital Mathematics Library.*, pages 77–84, Bertinoro, Italy. Masaryk University.

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*.

Minh Nghiem Quoc, Keisuke Yokoi, Yuichiroh Matsubayashi, and Akiko Aizawa. 2010. Mining coreference relations between formulas and text using wikipedia.

Marek Rei, Gamal K. O. Crichton, and Sampo Pyysalo. 2016. Attending to Characters in Neural Sequence Labeling Models. In *Coling 2016*.

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*.

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.

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.

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.