Parser lexicalisation through self-learning

Marek Rei
Computer Labratory
University of Cambridge
United Kingdom
Marek.Rei@cl.cam.ac.uk

Ted Briscoe
Computer Laboratory
University of Cambridge
United Kingdom
Ted.Briscoe@cl.cam.ac.uk

Abstract

We describe a new self-learning framework for parser lexicalisation that requires only a plain-text corpus of in-domain text. The method first creates augmented versions of dependency graphs by applying a series of modifications designed to directly capture higher-order lexical path dependencies. Scores are assigned to each edge in the graph using statistics from an automatically parsed background corpus. As bilexical dependencies are sparse, a novel directed distributional word similarity measure is used to smooth edge score estimates. Edge scores are then combined into graph scores and used for reranking the top-n analyses found by the unlexicalised parser. The approach achieves significant improvements on WSJ and biomedical text over the unlexicalised baseline parser, which is originally trained on a subset of the Brown corpus.

1 Introduction

Most parsers exploit supervised machine learning methods and a syntactically annotated dataset (i.e. treebank), incorporating a wide range of features in the training process to deliver competitive performance. The use of lexically-conditioned features, such as relations between lemmas or word forms, is often critical when choosing the correct syntactic analysis in ambiguous contexts. However, utilising such features leads the parser to learn information that is often specific to the domain and/or genre of the training data. Several experiments have demonstrated that many lexical features learnt in one domain provide little if any benefit when parsing text from different domains and genres (Sekine, 1997; Gildea, 2001). Furthermore, manual creation of in-domain treebanks is an expensive and time-consuming process, which can only be performed by experts with sufficient linguistic and domain knowledge.

In contrast, unlexicalised parsers avoid using lexical information and select a syntactic analysis using only more general features, such as POS tags. While they cannot be expected to achieve optimal performance when trained and tested in a single domain, unlexicalised parsers can be surprisingly competitive with their lexicalised counterparts (Klein and Manning, 2003; Petrov et al., 2006). In this work, instead of trying to adapt a lexicalised parser to new domains, we explore how bilexical features can be integrated effectively with any unlexicalised parser. As our novel self-learning framework requires only a large unannotated corpus, lexical features can be easily tuned to a specific domain or genre by selecting a suitable dataset. In addition, we describe a graph expansion process that captures selected bilexical relations which improve performance but would otherwise require sparse higher-order dependency path feature types in most approaches to dependency parsing. As many bilexical features will still be sparse, we also develop an approach to estimating confidence scores for dependency relations using a directional distributional word similarity measure. The final framework integrates easily with any unlexicalised (and therefore potentially less domain/genre-biased) parser capable of returning ranked dependency analyses.

2 Background

We hypothesise that a large corpus will often contain examples of dependency relations in non-ambiguous contexts, and these will mostly be correctly parsed by an unlexicalised parser. Lexical statistics derived from the corpus can then be used to select the correct parse in a more difficult context. For example, consider the following sentences:

(1) a. Government projects interest researchers

b. Government raises interest rates

c. Government projects receive funding

d. Interest rates are increasing

Noun-verb ambiguities over *projects* and *interest* might erroneously result in the unlexicalised parser returning similar dependency graphs for both *a* and *b*. However, sentences *c* and *d* contain less ambiguous instances of the same phrases and can provide clues to correctly parsing the first two examples. In a large in-domain corpus we are likely to find more cases of *researchers* being the object for *interest* and fewer cases where it is the object of *project*. In contrast, *rates* is more likely to have *interest* as a modifier than as a head in an object relation. Exploiting this lexical information, we can assign the correct derivation to each of the more ambiguous sentences.

Similar intuitions have been used to motivate the acquisition of bilexical features from background corpora for improving parser accuracy. However, previous work has focused on including these statistics as auxiliary features during supervised training. For example, van Noord (2007) incorporated bilexical preferences as features via self-training to improve the Alpino parser for Dutch. Plank and van Noord (2008) investigated the application of auxiliary distributions for domain adaptation. They incorporated information from both in-domain and out-of-domain sources into their maximum entropy model and found that the out-of-domain auxiliary distributions did not contribute to parsing accuracy in the target domain. Zhou et al. (2011) extracted n-gram counts from Google queries and a large corpus to improve the MSTParser. In contrast to previous work, we refer to our approach as self-learning because it differs from self-training by utilising statistics found using an initial parse ranking model to create a *separate* unsupervised reranking component, without retraining the baseline unlexicalised model.

We formulate our self-learning framework as a reranking process that assigns new scores to the top-n ranked analyses found by the original parser. Parse reranking has been successfully used in previous work as a method of including a wider range of features to rescore a smaller selection of highly-ranked candidate parses. Collins (2000) was one of the first to propose supervised reranking as an additional step to increase parser accuracy and achieved 1.55% accuracy improvement for his parser. Charniak and Johnson (2005) utilise a discriminative reranker and show a 1.3% improvement for the Charniak parser. McClosky et al. (2006) extend their work by adding new features and further increase the performance by 0.3%. Ng et al. (2010) implemented a discriminative maximum entropy reranker for the C&C parser and showed a 0.23% improvement over the baseline. Bansal and Klein (2011) discriminatively rerank derivations from the Berkeley unlexicalised parser (Petrov et al., 2006) demonstrating that lexical features derived from the Google n-gram corpus improve accuracy even when used in conjunction with other reranking features. They have all treated reranking as a supervised task and trained a discriminative classifier using parse tree features and annotated in-domain data. In contrast, our reranker only uses statistics from an unlabelled source and requires no manual annotation or training of the reranking component. As we utilise an unlexicalised parser, our baseline performance on WSJ text is lower compared to some fully-lexicalised parsers. However, an unlexicalised parser is also likely to be less biased to domains or genres manifested in the text used to train its original ranking model. This may allow the reranker to adapt it to a new domain and/or genre more effectively.

3 Reordering dependency graphs

For our experiments, we make use of the unlexicalised RASP parser (Briscoe et al., 2006) as the baseline system. For every sentence s the parser returns a list of dependency graphs G_s, ranked by the log probability of the associated derivation in the structural ranking model. Our goal is to reorder this

list to improve ranking accuracy and, most importantly, to improve the quality of the highest-ranked dependency graph. This is done by assigning a confidence score to every graph $g_{s,r} \in G_s$ where r is the rank of g_s for sentence s. The method treats each sentence independently, therefore we can omit the sentence identifiers and refer to $g_{s,r}$ as g_r.

We first calculate confidence scores for all the individual edges and then combine them into an overall score for the dependency graph. In the following sections, we describe a series of graph modifications that incorporates selected higher-order dependency path relations, without introducing unwanted noise or complexity into the reranker. Next, we outline different approaches for calculating and smoothing the confidence scores for bilexical relations. Finally, we describe methods for combining together these scores and calculating an overall score for a dependency graph. We make publically available all the code developed for performing these steps in the parse reranking system.[1]

3.1 Graph modifications

For every dependency graph g_r the graph expansion procedure creates a modified representation g'_r which contains a wider range of bilexical relations. The motivation for this graph expansion step is similar to that motivating the move from first-order to higher-order dependency path feature types (e.g., Carreras (2007)). However, compared to using all nth-order paths, these rules are chosen to maximise the utility and minimise the sparsity of the resulting bilexical features. In addition, the cascading nature of the expansion steps means in some cases the expansion captures useful 3rd and 4th order dependencies. Similar approaches to graph modifications have been successfully used for several NLP tasks (van Noord, 2007; Arora et al., 2010).

For any edge e we also use notation (rel, w_1, w_2), referring to an edge from w_1 to w_2 with the label rel. We perform the following modifications on every dependency graph:

1. **Normalising lemmas.** All lemmas are converted to lowercase. Numerical lemmas are replaced with more generic tags to reduce sparsity.

2. **Bypassing conjunctions.** For every edge pair (rel_1, w_1, w_2) and (rel_2, w_2, w_3) where w_2 is tagged as a conjunction, we create an additional edge (rel_1, w_1, w_3). This bypasses the conjunction node and creates direct edges between the head and dependents of the conjunctive lemma.

3. **Bypassing prepositions.** For every edge pair (rel_1, w_1, w_2) and (rel_2, w_2, w_3) where w_2 is tagged as a preposition, we create an additional edge (rel_3, w_1, w_3). $rel_3 = rel_1$ +'_prep', where '_prep' is added as a marker to indicate that the relation originally contained a preposition.

4. **Bypassing verbs.** For every edge pair (rel_1, w_1, w_2) and (rel_2, w_1, w_3) where w_1 is tagged as a verb, w_2 and w_3 are both tagged as open-class lemmas, rel_1 starts with a subject relation, and rel_2 starts with an object relation, we create an additional edge (rel_3, w_2, w_3) where $rel_3 = rel_1$ + '-' + rel_2. This creates an additional edge between the subject and the object, with the new edge label containing both of the original labels.

5. **Duplicating nodes.** For every existing node in the graph, containing the lemma and POS for each token (*lemma_pos*), we create a parallel node without the POS information (*lemma*). Then, for each existing edge, we create three corresponding edges, interconnecting the parallel nodes to each other and the original graph. This allows the reranker to exploit both specific and more generic instantiations of each lemma.

Figure 1 illustrates the graph modification process. It is important to note that each of these modifications gets applied in the order that they are described above. For example, when creating edges for bypassing verbs, the new edges for prepositions and conjunctions have already been created and also participate in this step. We performed ablation tests on the development data and verified that each of these modifications contributes positively to the final performance.

3.2 Edge scoring methods

We start the scoring process by assigning individual confidence scores to every bilexical relation in the

[1] www.marekrei.com/projects/lexicalisation

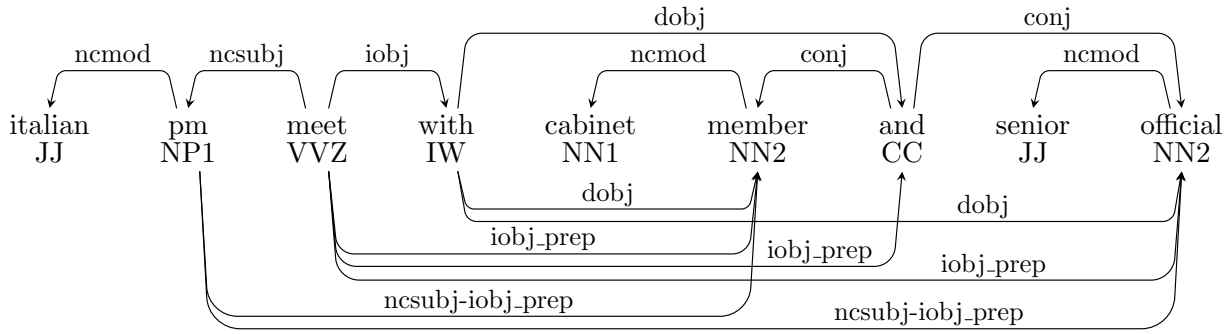

Figure 1: Modified graph for the sentence *'Italian PM meets with Cabinet members and senior officials'* after steps 1-4. Edges above the text are created by the parser, edges below the text are automatically created using the operations described in Section 3.1. The 5th step will create 9 new nodes and 45 additional edges (not shown).

modified graph. In this section we give an overview of some possible strategies for performing this task.

The parser returns a ranked list of graphs and this can be used to derive an edge score without requiring any additional information. We estimate that the likelihood of a parse being the best possible parse for a given sentence is roughly inversely proportional to the rank that it is assigned by the parser. These values can be summed for all graphs that contain a specific edge, normalised to approximate a probability. We then calculate the score for edge e as the Reciprocal Edge Score (RES) – the probability of e belonging to the best possible parse:

$$\text{RES}(e) = \frac{\sum_{r=1}^{R}[\frac{1}{r} \times \text{contains}(g'_r, e)]}{\sum_{r=1}^{R} \frac{1}{r}}$$

where R is the total number of parses for a sentence, and $\text{contains}(g'_r, e)$ returns 1 if graph g'_r contains edge e, and 0 otherwise. The value is normalised, so that an edge which is found in all parses will have a score of 1.0, but occurrences at higher ranks will have a considerably larger contribution.

The score of an edge can also be assigned by estimating the probability of that edge using a parsed reference corpus. van Noord (2007) improved overall parsing performance in a supervised self-training framework using feature weights based on pointwise mutual information:

$$\text{I}(e) = \log \frac{\text{P}(rel, w_1, w_2)}{\text{P}(rel, w_1, *) \times \text{P}(*, *, w_2)}$$

where $\text{P}(rel, w_1, w_2)$ is the probability of seeing an edge from w_1 to w_2 with label rel, $\text{P}(rel, w_1, *)$ is the probability of seeing an edge from w_1 to any node with label rel, and $\text{P}(*, *, w_2)$ is the probability of seeing any type of edge linking to w_2. Plank and van Noord (2008) used the same approach for semi-supervised domain adaptation but were not able to achieve similar performance benefits. In our implementation we omit the logarithm in the equation, as this improves performance and avoids problems with $\log(0)$ for unseen edges.

$\text{I}(e)$ compares the probability of the complete edge to the probabilities of partially specified edges, but it assumes that w_2 will have an incoming relation, and that w_1 will have an outgoing relation of type rel to some unknown node. These assumptions may or may not be true – given the input sentence, we have observed w_1 and w_2 but do not know what relations they are involved in. Therefore, we create a more general version of the measure that compares the probability of the complete edge to the individual probabilities of the two lemmas – the Conditional Edge Score (CES_1):

$$\text{CES}_1(e) = \frac{\text{P}(rel, w_1, w_2)}{\text{P}(w_1) \times \text{P}(w_2)}$$

where $\text{P}(w_1)$ is the probability of seeing w_1 in text, estimated from a background corpus using maximum likelihood.

Finally, we know that w_1 and w_2 are in a sentence together but cannot assume that there is a dependency relation between them. However, we can choose to think of each sentence as a fully connected graph, with an edge going from every lemma to every other lemma in the same sentence. If there exists

$$\text{ECES}_1(rel, w_1, w_2) = \frac{1}{2} \times \left(\frac{\sum_{c_1 \in C_1} \text{sim}(c_1, w_1) \times \frac{\text{P}(rel, c_1, w_2)}{\text{P}(c_1) \times \text{P}(w_2)}}{\sum_{c_1 \in C_1} \text{sim}(c_1, w_1)} + \frac{\sum_{c_2 \in C_2} \text{sim}(c_2, w_2) \times \frac{\text{P}(rel, w_1, c_2)}{\text{P}(w_1) \times \text{P}(c_2)}}{\sum_{c_2 \in C_2} \text{sim}(c_2, w_2)}\right)$$

$$\text{ECES}_2(rel, w_1, w_2) = \frac{1}{2} \times \left(\frac{\sum_{c_1 \in C_1} \text{sim}(c_1, w_1) \times \frac{\text{P}(rel, c_1, w_2)}{\text{P}(*, c_1, w_2)}}{\sum_{c_1 \in C_1} \text{sim}(c_1, w_1)} + \frac{\sum_{c_2 \in C_2} \text{sim}(c_2, w_2) \times \frac{\text{P}(rel, w_1, c_2)}{\text{P}(*, w_1, c_2)}}{\sum_{c_2 \in C_2} \text{sim}(c_2, w_2)}\right)$$

Figure 2: Expanded edge score calculation methods using the list of distributionally similar lemmas

no genuine relation between the lemmas, the edge is simply considered a $null$ edge. We can then find the conditional probability of the relation type given the two lemmas:

$$\text{CES}_2(e) = \frac{\text{P}(rel, w_1, w_2)}{\text{P}(*, w_1, w_2)}$$

where $\text{P}(rel, w_1, w_2)$ is the probability of the fully-specified relation, and $\text{P}(*, w_1, w_2)$ is the probability of there being an edge of any type from w_1 to w_2, including a $null$ edge. Using fully connected graphs, the latter is equivalent to the probability of w_1 and w_2 appearing in a sentence together, which again can be calculated from the background corpus.

3.3 Smoothing edge scores

Apart from RES, all the scoring methods from the previous section rely on correctly estimating the probability of the fully-specified edge, $\text{P}(rel, w_1, w_2)$. Even in a large background corpus these triples will be very sparse, and it can be useful to find approximate methods for estimating the edge scores.

Using smoothing techniques derived from work on language modelling, we could back-off to a more general version of the relation. For example, if *(dobj, read, publication)* is not frequent enough, the value could be approximated using the probabilities of *(dobj, read, *)* and *(dobj, *, publication)*. However, this can lead to unexpected results due to compositionality – while *(dobj, read, *)* and *(dobj, *, rugby)* can be fairly common, *(dobj, read, rugby)* is an unlikely relation.

Instead, we can consider looking at other lemmas which are similar to the rare lemmas in the relation. If *(dobj, read, publication)* is infrequent in the data, the system might predict that *book* is a reasonable substitute for *publication* and use *(dobj, read, book)* to estimate the original probability.

Given that we have a reliable way of finding likely substitutes for a given lemma, we can create expanded versions of CES_1 and CES_2, as shown in Figure 2. C_1 is the list of substitute lemmas for w_1, and $\text{sim}(c_1, w_1)$ is a measure showing how similar c_1 is to w_1. The methods iterate over the list of substitutes and calculate the CES score for each of the modified relations. The values are then combined by using the similarity score as a weight – more similar lemmas will have a higher contribution to the final result. This is done for both the head and the dependent in the original relation, and the scores are then normalised and averaged.

Experiments with a wide variety of distributional word similarity measures revealed that *WeightedCosine* (Rei, 2013), a directional similarity measure designed to better capture hyponymy relations, performed best. Hyponyms are more specific versions of a word and normally include the general properties of the hypernym, making them well-suited for lexical substitution. The *WeightedCosine* measure incorporates an additional directional weight into the standard cosine similarity, assigning different importance to individual features for the hyponymy relation. We retain the 10 most distributionally similar putative hyponyms for each lemma and substitute them in the relation. The original lemma is also included with similarity 1.0, thereby assigning it the highest weight. The lemma vectors are built from the same vector space model that is used for calculating edge probabilities, which includes all the graph modifications described in Section 3.1.

3.4 Combining edge scores

While the CES and ECES measures calculate confidence scores for bilexical relations using statistics from a large background corpus, they do not include any knowledge about grammar, syntax, or the con-

$$\text{CMB}_1(e) = \sqrt[3]{\text{RES}(e) * \text{CES}_1(e) * \text{CES}_2(e)} \qquad \text{CMB}_2(e) = \sqrt[3]{\text{RES}(e) * \text{ECES}_1(e) * \text{ECES}_2(e)}$$

Figure 3: Edge score combination methods

text in a specific sentence. In contrast, the RES score implicitly includes some of this information, as it is calculated based on the original parser ranking. In order to take advantage of both information sources, we combine these scores into CMB_1 and CMB_2, as shown in Figure 3.

3.5 Graph scoring

Every edge in graph g'_r is assigned a score indicating the reranker's confidence in that edge belonging to the best parse. We investigated different strategies for combining these values together into a confidence score for the whole graph. The simplest solution is to sum together individual edge scores, but this would lead to always preferring graphs that have a larger number of edges. Interestingly, averaging the edge scores does not produce good results either because it is biased towards smaller graph fragments containing only highly-confident edges.

We created a new scoring method which prefers graphs that cover all the nodes, but does not create bias for a higher number of edges. For every node in the graph, it finds the average score of all edges which have that node as a dependent. These scores are then averaged again over all nodes:

$$\text{NScore}(n) = \frac{\sum_{e \in E_g} \text{EdgeScore}(e) \times \text{isDep}(e, n)}{\sum_{e \in E_g} \text{isDep}(e, n)}$$

$$\text{GraphScore}(g) = \frac{\sum_{n \in N_g} \text{NScore}(n)}{|N_g|}$$

where g is the graph being scored, $n \in N_g$ is a node in graph g, $e \in E_g$ is an edge in graph g, $\text{isDep}(e, n)$ is a function returning 1.0 if n is the dependent in edge e, and 0.0 otherwise. $\text{NScore}(n)$ is set to 0 if the node does not appear as a dependent in any edges. We found this metric performs well, as it prefers graphs that connect together many nodes without simply rewarding a larger number of edges.

While the score calculation is done using the modified graph g'_r, the resulting score is directly assigned to the corresponding original graph g_r, and the reordering of the original dependency graphs is used for evaluation.

4 Experiments

4.1 Evaluation methods

In order to evaluate how much the reranker improves the highest-ranked dependency graph, we calculate the microaveraged precision, recall and F-score over all dependencies from the top-ranking parses for the test set. Following the official RASP evaluation (Briscoe et al., 2006) we employ the hierarchical edge matching scheme which aggregates counts up the dependency relation subsumption hierarchy and thus rewards the parser for making more fine-grained distinctions.[2] Statistical significance of the change in F-score is calculated by using the Approximate Randomisation Test (Noreen, 1989; Cohen, 1995) with 10^6 iterations.

We also wish to measure how well the reranker does at the overall task of ordering dependency graphs. For this we make use of an oracle that creates the perfect ranking for a set of graphs by calculating their individual F-scores; this ideal ranking is then compared to the output of our system. Spearman's rank correlation coefficient between the two rankings is calculated for each sentence and then averaged over all sentences. If the scores for all of the returned analyses are equal, this coefficient cannot be calculated and is set to 0.

4.2 DepBank

We evaluated our self-learning framework using the DepBank/GR reannotation (Briscoe and Carroll, 2006) of the PARC 700 Dependency Bank (King et al., 2003). The dataset is provided with the open-source RASP distribution[3] and has been used for evaluating different parsers, including RASP (Briscoe and Carroll, 2006; Watson et al., 2007) and

[2] Slight changes in the performance of the baseline parser compared to previous publications are due to using a more recent version of the parser and minor corrections to the gold standard annotation.

[3] ilexir.co.uk/2012/open-source-rasp-3-1/

C&C (Clark and Curran, 2007). It contains 700 sentences, randomly chosen from section 23 of the WSJ Penn Treebank (Marcus et al., 1993), divided into development (140 sentences) and test data (560 sentences). We made use of the development data to experiment with a wider selection of edge and graph scoring methods, and report the final results on the test data.

For reranking we collect up to 1000 top-ranked analyses for each sentence. The actual number of analyses that the RASP parser outputs depends on the sentence and can be smaller. As the parser first constructs parse trees and converts them to dependency graphs, several parse trees may result in identical graphs; we remove any duplicates to obtain a ranking of unique dependency graphs.

Our approach relies on a large unannotated corpus of in-domain text, and for this we used the BLLIP corpus containing 50M words of in-domain WSJ articles. Our version of this corpus excludes texts that are found in the Penn Treebank, thereby also excluding the section that we use for evaluation.

The baseline system is the unlexicalised RASP parser with default settings. In order to construct the upper bound, we use an oracle to calculate the F-score for each dependency graph individually, and then create the best possible ranking using these scores.

Table 1 contains evaluation results on the DepBank/GR test set. The baseline system achieves 76.41% F-score on the test data, with 32.70% average correlation. I and RES scoring methods give comparable results, with RES improving correlation by 9.56%. The CES and ECES scores all make use of corpus-based statistics and all significantly improve over the baseline system, with absolute increases in F-score of more than 2% for the fully-connected edge score variants.

Finally, we combine the RES score with the corpus-based methods and the fully-connected CMB_2 variant again delivers the best overall results. The final F-score is 79.21%, an absolute improvement of 2.8%, corresponding to 33.65% relative error reduction with respect to the upper bound. Correlation is also increased by 16.32%; this means the methods not only improve the chances of finding the best dependency graph, but also manage to create a better overall ranking. The F-scores for all the corpus-based scoring methods are statistically significant when compared to the baseline ($p < 0.05$).

By using our self-learning framework, we were able to significantly improve the original unlexicalised parser. To put the overall result in a wider perspective, Clark and Curran (2007) achieve an F-score of 81.86% on the DepBank/GR test sentences using the C&C lexicalised parser, trained on 40,000 manually-treebanked sentences from the WSJ. The unlexicalised RASP parser, using a manually-developed grammar and a parse ranking component trained on 4,000 partially-bracketed unlabelled sentences from a domain/genre balanced subset of Brown (Watson et al., 2007), achieves an F-score of 76.41% on the same test set. The method introduced here improves this to 79.21% F-score without using any further manually-annotated data, closing more than half of the gap between the performance of a fully-supervised in-domain parser and a more weakly-supervised more domain-neutral one.

We also performed an additional detailed analysis of the results and found that, with the exception of the auxiliary dependency relation, the reranking process was able to improve the F-score of all other individual dependency types. Complements and modifiers are attached with much higher accuracy, resulting in 3.34% and 3.15% increase in the corresponding F-scores. The non-clausal modifier relation (*ncmod*), which is the most frequent label in the dataset, increases by 3.16%.

4.3 Genia

One advantage of our reranking framework is that it does not rely on any domain-dependent manually annotated resources. Therefore, we are interested in seeing how it performs on text from a completely different domain and genre.

The GENIA-GR dataset (Tateisi et al., 2008) is a collection of 492 sentences taken from biomedical research papers in the GENIA corpus (Kim et al., 2003). The sentences have been manually annotated with dependency-based grammatical relations identical to those output by the RASP parser. However, it does not contain dependencies for all tokens and many multi-word phrases are treated as single units. For example, the tokens *'intracellular redox status'* are annotated as one node with label *intracellular_redox_status*. We retain this annotation and

	DepBank/GR				GENIA-GR			
	Prec	**Rec**	**F**	ρ	**Prec**	**Rec**	**F**	ρ
Baseline	77.91	74.97	76.41	32.70	79.91	78.86	79.38	36.54
Upper Bound	86.74	82.82	84.73	75.36	86.33	84.71	85.51	78.66
I	77.77	75.00	76.36	33.32	77.18	76.21	76.69	30.23
RES	78.13	74.94	76.50	42.26	80.06	78.89	79.47	**47.52**
CES_1	79.68	76.40	78.01	41.95	78.64	77.50	78.07	36.06
CES_2	80.48	77.28	78.85	48.43	79.92	78.92	79.42	43.09
$ECES_1$	79.96	76.68	78.29	42.41	79.09	78.11	78.60	38.02
$ECES_2$	80.71	77.52	79.08	**49.05**	79.84	78.95	79.39	43.64
CMB_1	80.64	77.31	78.94	48.25	80.60	79.51	80.05	44.96
CMB_2	**80.88**	**77.60**	**79.21**	49.02	**80.69**	**79.64**	**80.16**	46.24

Table 1: Performance of different edge scoring methods on the test data. For each measure we report precision, recall, F-score, and average Spearman's correlation (ρ). The highest results for each measure are marked in bold. The underlined F-scores are significantly better compared to the baseline.

allow the unlexicalised parser to treat these nodes as atomic unseen words during POS tagging and parsing. However, we use the last lemma in each multi-word phrase for calculating the edge score statistics.

In order to initialise our parse reranking framework, we also need a background corpus that closely matches the evaluation domain. The annotated sentences in GENIA-GR were chosen from abstracts that are labelled with the MeSH term *'NF-kappa B'*. Following this method, we created our background corpus by extracting 7,100 full-text articles (1.6M sentences) from the PubMed Central Open Access collection, containing any of the following terms with any capitalisation: *'nf-kappa b'*, *'nf-kappab'*, *'nf kappa b'*, *'nf-kappa_b'*, *'nf-kb'*, *'nf-κb'*. Since we retain all texts from matching documents, this keyword search acts as a broad indicator that the sentences contain topics which correspond to the evaluation dataset. This focussed corpus was then parsed with the unlexicalised parser and used to create a statistical model for the reranking system, following the same methods as described in Sections 3 and 4.2.

Table 1 also contains the results for experiments in the biomedical domain. The first thing to notice is that while the upper bound for the unlexicalised parser is similar to that for the DepBank experiments in Section 4.2, the baseline results are considerably higher. This is largely due to the nature of the dataset – since many complicated multi-word phrases are treated as single nodes, the parser is not evaluated on edges within these nodes. In addition, treating these nodes as unseen words eliminates many incorrect derivations that would otherwise split the phrases. This results in a naturally higher baseline of 79.38%, and also makes it more difficult to further improve the performance.

The edge scoring methods I, CES_1 and $ECES_1$ deliver F-scores lower than the baseline in this experiment. RES, CES_2 and $ECES_2$ yield a modest improvement in both F-score and Spearman's correlation. Finally, the combination methods again give the best performance, with CMB_2 delivering an F-score of 80.16%, an absolute increase of 0.78%, which is statistically significant ($p < 0.05$). The experiment shows that our self-learning framework works on very different domains, and it can be used to significantly increase the accuracy of an unlexicalised parser without requiring any annotated data.

5 Conclusion

We developed a new self-learning framework for dependency graph reranking that requires only a plain-text corpus from a suitable domain. We automatically parse this corpus and use the highest ranked analyses to estimate maximum likelihood probabilities for bilexical relations. Every dependency graph is first modified to incorporate additional edges that model selected higher-order dependency path relationships. Each edge in the graph is then assigned a confidence score based on statistics from the background corpus and ranking preferences from the un-

lexicalised parser. We also described a novel method for smoothing these scores using directional distributional similarity measures. Finally, the edge scores are combined into an overall graph score by first averaging them over individual nodes.

As the method requires no annotated data, it can be easily adapted to different domains and genres. Our experiments showed that the reranking process significantly improved performance on both WSJ and biomedical data.

References

Shilpa Arora, Elijah Mayfield, Carolyn Penstein-Rosé, and Eric Nyberg. 2010. Sentiment Classification using Automatically Extracted Subgraph Features. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text.*

Mohit Bansal and Dan Klein. 2011. Web-scale features for full-scale parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 693–702.

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the COLING/ACL on Main conference poster sessions*, number July, pages 41–48, Morristown, NJ, USA. Association for Computational Linguistics.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, number July, pages 77–80, Sydney, Australia. Association for Computational Linguistics.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, volume 7, pages 957–961.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05*, 1(June):173–180.

Stephen Clark and James R. Curran. 2007. Formalism-independent parser evaluation with CCG and DepBank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, volume 45, pages 248–255.

Paul R Cohen. 1995. *Empirical Methods for Artificial Intelligence*. The MIT Press, Cambridge, MA.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *The 17th International Conference on Machine Learning (ICML)*.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(1):180–182.

Tracy H. King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 dependency bank. In *Proceedings of the EACL03: 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, pages 1–8.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, number July, pages 423–430. Association for Computational Linguistics Morristown, NJ, USA.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, pages 1–22.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, number June, pages 152–159, Morristown, NJ, USA. Association for Computational Linguistics.

Dominick Ng, Matthew Honnibal, and James R. Curran. 2010. Reranking a wide-coverage CCG parser. In *Australasian Language Technology Association Workshop 2010*, page 90.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses: An Introduction*. Wiley, New York.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL (ACL '06)*, pages 433–440, Morristown, NJ, USA. Association for Computational Linguistics.

Barbara Plank and Gertjan van Noord. 2008. Exploring an auxiliary distribution based approach to domain adaptation of a syntactic disambiguation model. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross- Domain Parser Evaluation*, pages 9–16, Manchester, UK. Association for Computational Linguistics.

Marek Rei. 2013. *Minimally supervised dependency-based methods for natural language processing*. Ph.D. thesis, University of Cambridge.

Satoshi Sekine. 1997. The domain dependence of parsing. In *Proceedings of the fifth conference on Applied natural language processing*, volume 1, pages 96–102, Morristown, NJ, USA. Association for Computational Linguistics.

Yuka Tateisi, Yusuke Miyao, Kenji Sagae, and Jun'ichi Tsujii. 2008. GENIA-GR: a Grammatical Relation Corpus for Parser Evaluation in the Biomedical Domain. In *Proceedings of LREC*, pages 1942–1948.

Gertjan van Noord. 2007. Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of the 10th International Conference on Parsing Technologies*, number June, pages 1–10, Morristown, NJ, USA. Association for Computational Linguistics.

Rebecca Watson, Ted Briscoe, and John Carroll. 2007. Semi-supervised training of a statistical parser from unlabeled partially-bracketed data. *Proceedings of the 10th International Conference on Parsing Technologies - IWPT '07*, (June):23–32.

Guangyou Zhou, Jun Zhao, Kang Liu, and Li Cai. 2011. Exploiting Web-Derived Selectional Preference to Improve Statistical Dependency Parsing. In *49th Annual Meeting of the Association for Computational Linguistics*, pages 1556–1565.