Improved CCG Parsing with Semi-supervised Supertagging

Mike Lewis
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
mike.lewis@ed.ac.uk

Mark Steedman
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
steedman@inf.ed.ac.uk

Abstract

Current supervised parsers are limited by the size of their labelled training data, making improving them with unlabelled data an important goal. We show how a state-of-the-art CCG parser can be enhanced, by predicting lexical categories using unsupervised vector-space embeddings of words. The use of word embeddings enables our model to better generalize from the labelled data, and allows us to accurately assign lexical categories without depending on a POS-tagger. Our approach leads to substantial improvements in dependency parsing results over the standard supervised CCG parser when evaluated on Wall Street Journal (0.8%), Wikipedia (1.8%) and biomedical (3.4%) text. We compare the performance of two recently proposed approaches for classification using a wide variety of word embeddings. We also give a detailed error analysis demonstrating where using embeddings outperforms traditional feature sets, and showing how including POS features can decrease accuracy.

1 Introduction

Combinatory Categorial Grammar (CCG) is widely used in natural language semantics (Bos, 2008; Kwiatkowski et al., 2010; Krishnamurthy and Mitchell, 2012; Lewis and Steedman, 2013a; Lewis and Steedman, 2013b; Kwiatkowski et al., 2013), largely because of its direct linkage of syntax and semantics. However, this connection means that performance on semantic applications is highly dependent on the quality of the syntactic parse. Although CCG parsers perform at state-of-the-art levels (Rimell et al., 2009; Nivre et al., 2010), full-sentence accuracy is just 25.6% on Wikipedia text, which gives a low upper bound on logical inference approaches to question-answering and textual entailment.

Supertags are rich lexical categories that go beyond POS tags by encoding information about predicate-argument structure. Supertagging is "almost parsing", and is used by parsers based on strongly lexicalized formalisms such as CCG and TAG to improve accuracy and efficiency, by delegating many of the parsing decisions to finite-state models (Bangalore and Joshi, 1999). A disadvantage of this approach is that larger sets of lexical categories mean increased sparsity, decreasing tagging accuracy. As large amounts of labelled data are unlikely to be made available, recent work has explored using unlabelled data to improve parser lexicons (Thomforde and Steedman, 2011; Deoskar et al., 2011; Deoskar et al., 2014). However, existing work has failed to improve the overall accuracy of state-of-the-art supervised parsers in-domain.

Another strand of recent work has explored using unsupervised word embeddings as features in supervised models (Turian et al., 2010; Collobert et al., 2011b), largely motivated as a simpler and more general alternative to standard feature sets. We apply similar techniques to CCG supertagging, hypothesising that words which are close in the embedding space will have similar supertags. Most existing work has focused on flat tagging tasks, and has not produced state-of-the-art results on structured prediction tasks like parsing (Collobert, 2011; Andreas and Klein, 2014). CCG's lexicalized nature provides a simple and elegant solution to treating parsing as a flat tagging task, as the lexical categories encode information about hierarchical structure.

Transactions of the Association for Computational Linguistics, 2 (2014) 327–338. Action Editor: Ryan McDonald.
Submitted 4/2014; Revised 6/2014; Published 10/2014.

As well as improving parsing accuracy, our model has a number of advantages over current CCG parsing work. Our supertagger does not make use of a POS-tagger, a fact which simplifies the model architecture, reduces the number of parameters, and eliminates errors caused by a pipeline approach. Also, learning word embeddings is an active area of research, and future developments may directly lead to better parsing accuracy, with no change required to our model.

2 Background

2.1 CCG Parsing

The widely-used C&C parser (Clark and Curran, 2007) for CCG takes a pipeline approach, where first sentences are POS-tagged, then supertagged, and then parsed. The supertagger outputs a distribution over tags for each word, and a beam is used to aggressively prune supertags to reduce the parser search space. If the parser is unable to find a parse with a given set of supertags, the beam is relaxed. This approach is known as *adaptive supertagging*.

The pipeline approach has two major drawbacks. Firstly, the use of a POS-tagger can overly prune the search space for the supertagger. Whilst POS-taggers have an accuracy of around 97% in domain, this drops to just 93.4% on biomedical text (Rimell and Clark, 2009), meaning that most sentences will contain an erroneous POS-tag. The supertagger model is overly dependent on POS-features—in Section 4.6 we show that supertagger performance drops dramatically on words which have been assigned an incorrect POS-tag.

Secondly, both the POS-tagger and supertagger are highly reliant on lexical features, meaning that performance drops both on unknown words, and words used differently from the training data. Many common words do not appear at all in the training data of the Penn Treebank, such as *ten*, *militants*, *insight*, and *teenager*. Many others are not seen with all their possible uses—for example *European* only occurs as an adjective, never a noun, meaning that the C&C parser is unable to analyse simple sentences like *The director of the IMF is traditionally a European*. These problems are particularly acute when parsing other domains (Rimell and Clark, 2009).

2.2 Semi Supervised NLP using Word Embeddings

Recent work has explored using vector space embeddings for words as features in supervised models for a variety of tasks, such as POS-tagging, chunking, named-entity recognition, semantic role labelling, and phrase structure parsing (Turian et al., 2010; Collobert et al., 2011b; Collobert, 2011; Socher et al., 2013). The major motivation for using these techniques has been to minimize the level of task-specific feature engineering required, as the same feature set can lead to good results on a variety of tasks. Performance varies between tasks, but any gains over state-of-the-art traditional features have been small. A variety of techniques have been used for learning such embeddings from large unlabelled corpora, such as neural-network language models.

3 Models

We introduce models for predicting CCG lexical categories based on vector-space embeddings. The models can then be used to replace the POS-tagging and supertagging stages used by existing CCG parsers. We experiment with the neural network model proposed by Collobert et al. (2011b), and conditional random field (CRF) model used by Turian et al. (2010). We only use features that can be expected to work well out-of-domain—in particular, we use no lexical or POS features.

3.1 Features

Our features are similar to those used by Collobert et al. (2011b) for POS-tagging. For every word in a context window, we add features for the embedding of the word, its 2-character suffix, and whether or not it is capitalised. We expect such features to generalize well to other domains—and in Section 4.5 we show that adding traditional POS-tag and lexical features does not help.

To further reduce sparsity, we apply some simple preprocessing techniques. Words are lower-cased[1], and all digits are replaced with *0*. If an unknown word is hyphenated, we first try backing-off to the substring after the hyphen.

[1]For embeddings that include separate entries for the same word with different capitalization, we take the most frequently occurring version in the unlabelled corpus.

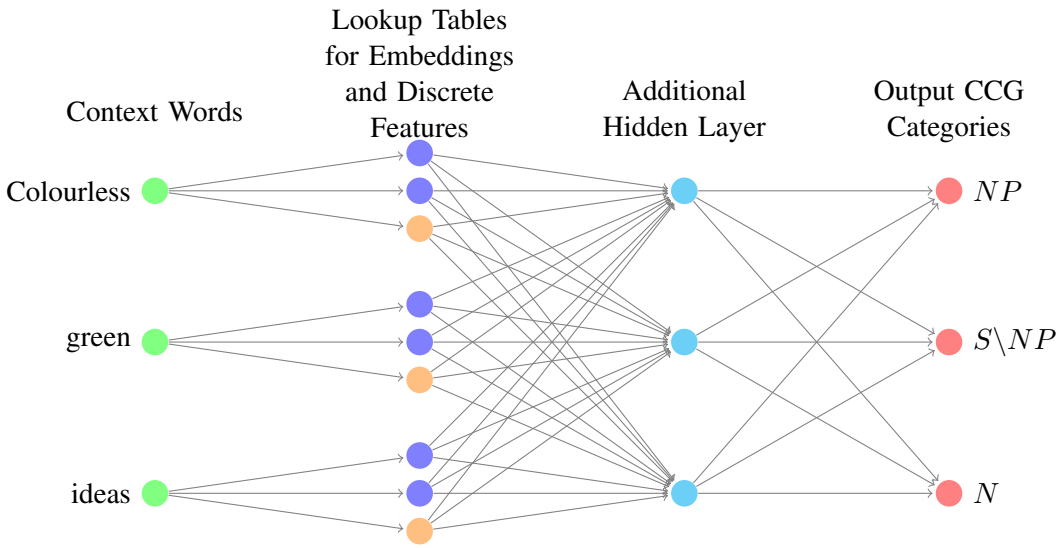

Figure 1: Collobert et al. (2011b)'s *Window approach network*, applied to CCG supertagging. Each context word C_i is connected to one hidden node per dimension of the embedding E_{ij} with weight W_{ij}, and additional hidden nodes representing capitalization and suffix features. The weights W_{ij} are initialized using pre-trained word embeddings, but can be modified during supervised training. The additional hidden layer uses a *hard-tanh* activation function, and the output layer uses a *softmax* activation function.

Words which do not have an entry in the word embeddings share an 'unknown' embedding. Different 'unknown' vectors are used for capitalized and uncapitalized words, and non-alphabetic symbols. We also add entries for context words which are before the start and after the end of the sentence. All of these were initialized to the 'unknown' vector in the pre-trained embeddings (or with Gaussian noise if not available).

3.2 Neural Network Model

We predict word supertags with the neural network classifier used by Collobert et al. (2011b) for POS-tagging, as shown in Figure 1. Each feature is implemented as a lookup table, which maps context words onto vectors. The same lookup table parameters are used wherever a word appears in the context window.

Word embeddings are implemented with a lookup table $W \in R^{V \times D}$, where V is the size of the vocabulary, and D is the dimension of the word embeddings. The parameters of the lookup table are initialized using unsupervised embeddings, but are modified during supervised training.

As in Collobert et al. (2011b), non-embedding features (2-character suffixes and capitalization) are also each represented with lookup tables, which map each feature onto a K dimensional vector (as in Collobert et al. (2011b), we use $K = 5$). Lookup table parameters for non-embeddings features are initialized with Gaussian noise.

The first hidden layer therefore contains $C \times (D + KF)$ nodes, where F is the number of discrete features and C is the size of the context window. We also experimented adding an additional hidden layer, with a *hard-tanh* activation function, which makes the classifier non-linear. Finally, a *logistic softmax* layer is used for classifying output categories.

The model is trained using stochastic gradient descent, with a learning rate of 0.01, optimizing for cross-entropy. We use early-stopping as an alternative to regularization—after each iteration the model is evaluated for accuracy on held-out data, and we use the best performing model. Training was run until performance decreased on held-out data.

3.3 CRF Model

The neural network model treats the probability of each supertag as being conditionally independent. However, conditioning on surrounding supertags

Embeddings	Model	Dimensionality	Training Words	Training Domain
Collobert&Weston	NNLM	50	660M	Wikipedia
Skip-Gram	Skip Gram	300	100B	Google News
Turian	NNLM	25, 50, 100, 200	37M	Newswire
HLBL	HLBL	50, 100	37M	Newswire
Mikolov	RNNLM	80, 640	320M	Broadcast News

Table 1: Embeddings used in our experiments. Dimensionality is the set of dimensions of the word embedding space that we experimented with, and Training Words refers to the size of the unlabelled corpus the embeddings were trained on.

may be very useful—for example, a noun is much more likely to follow an adjective than a verb. Curran et al. (2006) report a large improvement using a maximum-entropy Markov model for supertagging, conditioned on the surrounding supertags.

We follow Turian et al. (2010) in using a linear chain CRF model for sequence classification using embeddings as features. This model does not allow supervised training to fine-tune the embeddings, though it would be possible to build a CRF/NN hybrid that enabled this. We use the same feature set as with the neural network model—so the probability of a category depends on embeddings, capitalization and suffix features—as well as the previous category. The model is trained using the averaged-perceptron algorithm (Collins, 2002), again using early-stopping based on development data accuracy.

4 Experiments

4.1 Domains

We experiment with three domains:

- CCGBank (Hockenmaier and Steedman, 2007), which is a conversion of the Penn Treebank (Marcus et al., 1993) to CCG. Section 23 is used for evaluation.

- Wikipedia, using the corpus of 200 sentences annotated with CCG derivations by Honnibal et al. (2009). As the text is out-of-domain, parsing accuracy drops substantially on this corpus.

- Biomedical text, which is even less related to the newswire text than Wikipedia, due to large numbers of unseen words and different writing styles, causing low parsing accuracy. For parsing experiments, we use the Bioinfer corpus (Pyysalo et al., 2007) as a test set. For measuring supertagging accuracy, we use the CCG annotation produced by Rimell and Clark (2008).

4.2 Neural Network Model Parameters

In this section, we explore how adjusting the parameters of our neural network model[2] affects 1-best lexical category accuracy on the Section 00 of CCGBank (all development was done on this data). The C&C supertagger achieves 91.5% accuracy on this task. The models were trained on Sections 02-21 of CCGBank, and the reported numbers are the best accuracy achieved on Section 00. As in Clark and Curran (2007), all models use only the 425 most frequent categories in CCGBank.

4.2.1 Embeddings

A number of word embeddings have recently been released, aiming to capture a variety of syntactic and semantic phenomena, based on neural network language models (NNLMs) (Turian et al., 2010; Collobert et al., 2011b), recurrent neural network language models (Mikolov, 2012), the hierarchical log bilinear model (HLBL) (Mnih and Hinton, 2008), and Mikolov et al. (2013)'s Skip Gram model. However, there has been a lack of experiments comparing which embeddings provide the most effective features for downstream tasks.

First, we investigated the performance of several publicly available embeddings, to find which was most effective for supertagging. The embeddings we used are summarized in Table 1. For efficiency, we used our simplest architecture, with no additional

[2]Implemented using the Torch7 library (Collobert et al., 2011a)

Embeddings	Category Accuracy (window=5)	Category Accuracy (window=7)
Collobert&Weston	90.0%	89.6%
Skip Gram	90.9%	91.0%
Turian-25	91.0%	91.1%
Turian-50	**91.1%**	**91.3%**
Turian-100	91.0%	91.1%
Turian-200	90.8%	90.7%
HLBL-50	90.9%	91.2%
HLBL-100	**91.1%**	**91.3%**
Mikolov-80	87.2%	88.1%
Mikolov-640	87.9%	88.4%

Table 2: Comparison of different embeddings and context windows on Section 00 of CCGBank. Abbreviations such as Turian-50 refer to the Turian embeddings with a 50-dimensional embedding space.

hidden layer. We also investigate which size context window is most effective.

Results are shown in Table 2, and show that the choice of embeddings is crucial to performance on this task. The performance of the Turian and HLBL embeddings is surprisingly high given the relatively small amount of unlabelled data, suggesting that parameters other than the size of the corpus are more important. Of course, we have not performed a grid-search of the parameter space, and it is possible that other embeddings would perform better with different training data, dimensionality, or model architectures. The Mikolov embeddings may suffer from being trained on broadcast news, which has no punctuation and different language use. Using a context window of 7 words generally outperformed using a window of 5 words (we also experimented with a 9 word window, but found performance decreased slightly to 91.2% for the Turian-50 embeddings). There is no clear trend on the optimal dimension of the embedding space, and it is likely to vary with training methods and corpus size.

Next, we experimented with the size of the additional hidden layer—for efficiency, using the Turian-50 embeddings with a 5-word context window. Results are shown in Table 3, and suggest that a hidden layer is not useful for this task—possibly due to over-fitting.

Embeddings	Additional Hidden Layer Size	Accuracy
Turian-50	0	91.1%
Turian-50	100	90.9%
Turian-50	300	90.6%
Turian-50	500	90.9%
Turian-50	1000	90.5%

Table 3: Comparison of different model architectures, using the Turian embeddings and a 5-word context window. A size of 0 means no additional hidden layer was used.

In all subsequent experiments we used a context window of 7 words, no additional hidden layer, and the Turian-50 embeddings.

4.3 CRF Model

We also experimented with the CRF model for supertagging[3]. Training these models took far longer than our neural-network model, due to the need to use the forward-backward algorithm with a 425×425-dimensional transition matrix during training (rather than considering each word's category independently). Consequently, we only experimented with the Turian-50 embeddings with a 7-word context window, which attained the best performance using the neural network.

We found that using the Turian-50 embeddings gave a surprisingly weak performance of just 90.3% (compared to 91.3% for the neural network model). We hypothesised that one reason for this result could be that the model is unable to modify the embeddings during supervised training (in contrast to the neural-network model). Consequently, we built a new set of embeddings, using the weight-matrix learned by our best neural network model. A new CRF model was then trained using the tuned embeddings. Performance then improved dramatically to 91.5%, and slightly outperformed the neural network—showing that while there is a small advantage to using sequence information, it is crucial to allow supervised training to modify the embeddings. These results help explain why Collobert et al. (2011b)'s neural network models outperform Turian et al. (2010)'s sequence models—but greater

[3]Implemented using CRFSuite (Okazaki,)

improvements may be possible with the combined approach we introduce here, which allows the model to both tune the embeddings and exploit sequence information. However, tagging with this model was considerably slower than the neural network (again, due to the cost of decoding), so we used the neural network architecture in the remaining experiments.

4.4 Multitagging Accuracy

The C&C parser takes a set of supertags per word as input, which is used to prune the search space. If no parse is found, the sentence is supertagged again with a wider beam. The effectiveness of the pruning therefore depends on the accuracy of the supertagger at a given level of ambiguity.

We experimented with the accuracy of different supertaggers at different levels of ambiguity. For the C&C supertagger, we vary the number of categories per word using the same back-off beam settings reported in Clark and Curran (2007). For our supertagger, we vary recall by adjusting the a variable-width beam, which removes tags whose probability is less than β times that of the most likely tag.

Results are shown in Figure 2. The supertaggers based on embeddings consistently match or outperform the C&C supertagger at all levels of recall across all domains. While performance is similar with a small number of tags per word, our supertaggers perform better with a more relaxed beam—perhaps representing cases which are challenging for the C&C model, such as POS-tag errors.

4.5 Parsing Accuracy

We investigate whether our supertagger improves the performance of the C&C parser, by replacing the standard C&C supertagger with our model. This evaluation is somewhat indirect, as the parser does not make use of the supertagger probabilities for categories, but instead simply uses it to prune the search space. However, we show that better pruning leads directly to better parsing accuracies.

C&C parser results on CCGBank and Wikipedia are reported using Clark and Curran (2007)'s best performing *hybrid* model[4] (trained on Sections 02-21), with automatic POS-tags, and the parameters

[4]This model is not publicly available, so we re-trained it following the instructions at `http://aclweb.org/aclwiki/index.php?title=Training_the_C&C_Parser`

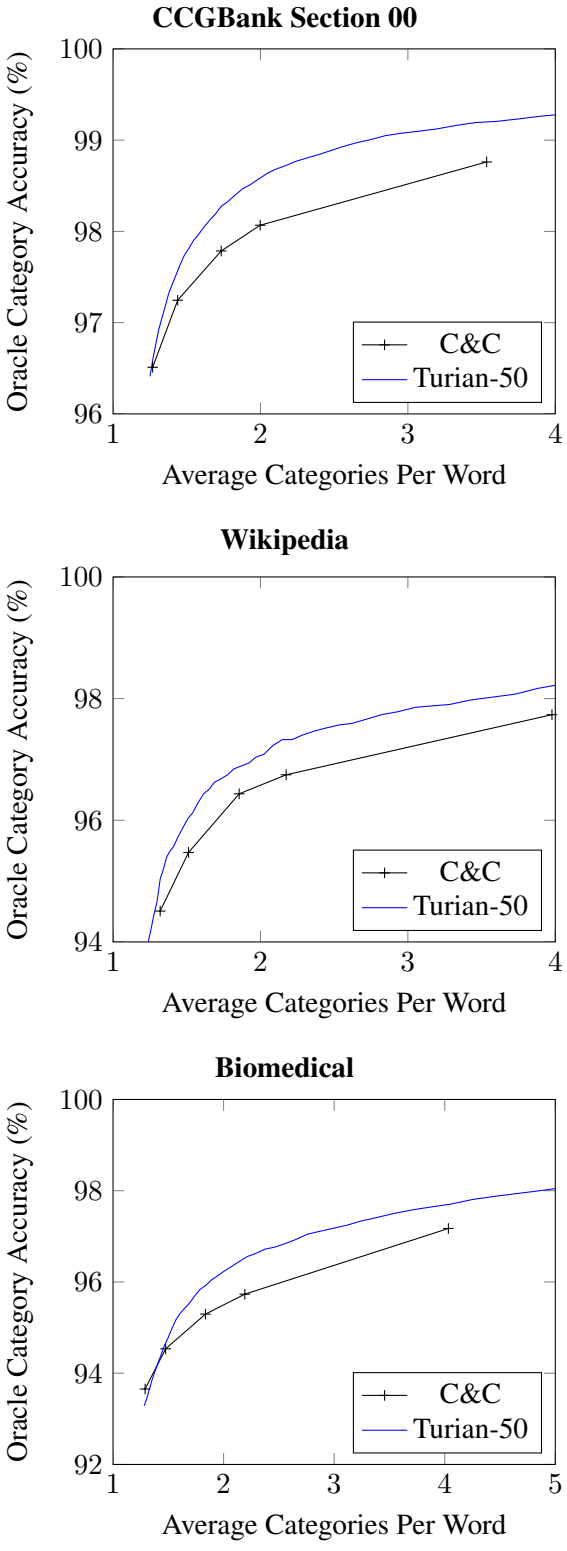

Figure 2: Ambiguity vs. Accuracy for different supertaggers across different domains. Datapoints for the C&C parser use its standard back-off parameters.

Supertagger	CCGBank			Wikipedia			Bioinfer		
	F1 (cov)	COV	F1 (all)	F1 (cov)	COV	F1 (all)	F1 (cov)	COV	F1 (all)
C&C	85.47	99.6	85.30	81.19	99.0	80.64	76.08	97.2	74.88
Honnibal et al. (2009)	85.19	99.8	-	81.75	99.4	-	-	-	-
Brown Clusters	85.27	99.9	85.21	80.89	**100.0**	80.89	76.06	**100.0**	76.06
Turian-50 Embeddings	**86.11**	**100.0**	**86.11**	82.30	**100.0**	82.30	**78.41**	99.8	**78.28**
Turian-50 + POS tags	85.62	99.9	85.55	81.77	**100.0**	81.77	77.05	**100.0**	77.05
Turian-50 + Frequent words	86.04	**100.0**	86.04	**82.44**	**100.0**	**82.44**	78.10	**100.0**	78.10

Table 4: Parsing F1-scores for labelled dependencies across a range of domains, using the C&C parser with different supertaggers. Embeddings models used a context window of 7 words, and no additional hidden layer. Following previous CCG parsing work, we report F1-scores on the subset of sentences where the parser is able to produce a parse (F1-cov), and the parser's coverage (COV). Where available we also report overall scores (F1-all), including parser failures, which we believe gives a more realistic assessment.

used in the published results. Biomedical results use the publicly available parsing model, setting the 'parser beam ratio' parameter to 10^{-4}, which improved results on development data. To achieve full coverage on the Wikipedia corpus, we increased the 'max supercats' parameter to 10^7. C&C accuracies differ very slightly from previously reported results, due to differences in the retrained models.

As in Clark and Curran (2007), we use a variable-width beam β that prunes categories whose probability is less than β times that of the most likely category. For simplicity, our supertaggers use the same β back-off parameters as are used by the C&C parser, though it is possible that further improvements could be gained by carefully tuning these parameters.[5] In contrast to the C&C supertaggers, we do not make use a tag-dictionaries.

Results are shown in Table 4, and our supertaggers consistently lead to improvements over the baseline parser across all domains, with larger improvements out-of-domain. Our best model also outperforms Honnibal et al. (2009)'s self-training approach to domain adaptation on Wikipedia (which lowers performance on CCGBank).

Our results show that word embeddings are an effective way of adding distributional information into CCG supertagging. A popular alternative approach for semi-supervised learning is to use Brown clusters (Brown et al., 1992). To ensure a fair comparison with the Turian embeddings, we use clusters trained on the same corpus, and use a comparable feature set (clusters, capitalization, and 2-character suffixes—all implemented as sparse binary features). Brown clusters are hierarchical, and following Koo et al. (2008), we incorporate Brown clusters features at multiple levels of granularity—using 64 coarse clusters (loosely analogous to POS-tags) and 1000 fine-grained clusters. Results show slightly lower performance than C&C in domain, but higher performance out of domain. However, they are substantially lower than results using the Turian-50 embeddings.

We also experimented with adding traditional word and POS features, which were implemented as sparse vectors for each word in the context window. We found that including POS features (derived from the C&C POS-tagger) reduced accuracy across all domains. One reason is that POS tags are highly discriminative features, therefore errors can be hard to recover from. Adding lexical features for the most frequent 250 words had little impact on results, showing that the embeddings already represent this information.

For infrequent words, the C&C parser uses a hard constraint that only certain POS-tag/supertag combinations are allowed. This constraint means that the parser may be particularly vulnerable to POS-

[5]We briefly experimented setting the β parameters to match the ambiguity of the C&C supertagger on Section 00 of CCGBank, which caused the F1-score using the Turian-50 embeddings to drop slightly from 86.11 to 85.95.

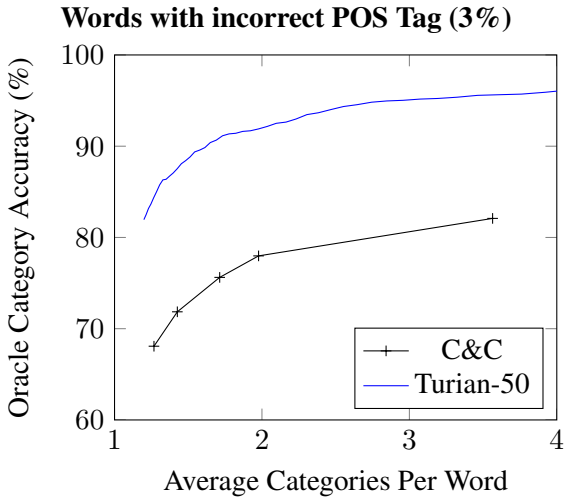

Figure 3: Ambiguity vs. Accuracy for different supertaggers, on words with incorrect POS tags.

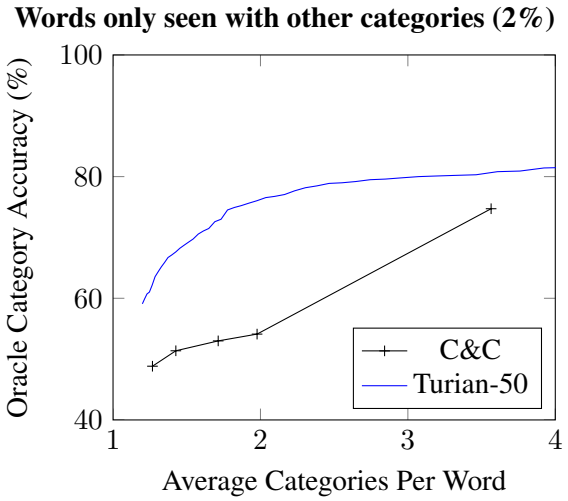

Figure 4: Ambiguity vs. Accuracy for different supertaggers, on words that do occur in the training data, but not with the category required in the test data.

tag errors, as the model cannot override the hard-constraint. We therefore also ran the model allowing any POS-tag/supertag combination. We found that parsing accuracy was 0.02 higher on development data (and much slower), suggesting that the model itself is overly reliant on POS-features.

4.6 Error Analysis

We have demonstrated that word embeddings are highly effective for CCG supertagging. In this section, we investigate several cases in which they are particularly helpful—by measuring supertagger performance when the POS tagger made mistakes, when words were unseen in the labelled data, and when the labelled data only contains the word with a different category. Our supertaggers show substantial improvements over more complex existing models.

Figure 3 shows performance when the POS-tagger assigns the wrong tag to a word. Both systems show dramatically lower performance on these cases—the embeddings supertagger does not use POS features, but POS errors are likely to represent generally difficult examples. However, the embeddings supertagger is almost 15% more accurate on this subset than the C&C supertagger, and with a relaxed beam reaches 96% accuracy, showing the advantages of avoiding a pipeline approach. In contrast, the C&C tagger is not robust to POS tagger errors, and asymptotes at just 82% accuracy. An alternative way of mitigating POS errors is to use a distribution over POS tags as features in the supertagger—Curran et al. (2006) show that this technique improves supertagging accuracy by 0.4% over the C&C baseline, but do not report the impact on parsing.

Figure 4 shows performance when the a word has been seen in the training data, but only with a different category from the instance in the test data (for example, *European* only occurs as a adjective in the training data, but it may occur as a noun in the test data). Performance is even worse on these cases, which appear to be extremely difficult for existing models. The accuracy of the embeddings supertagger converges at just 80%, suggesting that our model has overfit the labelled data. However, it still outperforms the C&C supertagger by 22% with a beam allowing 2 tags per word. The large jump in C&C supertagger performance for the final back-off level is due to a change in the word frequency threshold at which the C&C parser only considers word/category pairs that occur in the labelled data.

Figure 5 gives results for cases where the word is unseen in the labelled data. The C&C supertagger performance is surprisingly good on such cases, suggesting that the morphological and context used

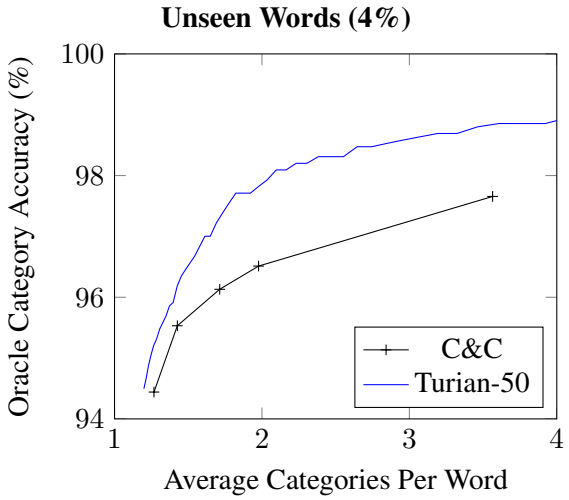

Figure 5: Ambiguity vs. Accuracy for different supertaggers, on words which are unseen in the labelled data.

is normally sufficient for inferring categories for unseen words. However, our supertagger still clearly outperforms the C&C supertagger, suggesting that the large vocabulary of the unsupervised embeddings helps it to generalize from the labelled data.

We also investigated supertagging accuracy on different types of word—Table 5 shows several interesting cases. While performance on nouns is similar, our supertagger is substantially better on verbs. Verbs can have many different CCG categories, depending on the number of arguments and tense, and not all valid word/category combinations will be seen in the labelled data. Our embeddings allow the supertagger to learn generalizations, such as that transitive verbs can also often have intransitive uses. Similarly, wh-words can have many possible categories in complex constructions like relative clauses and pied-piping—and our embeddings may help the model generalize from having seen a category for *which* to one for *whom*. On the other hand, the C&C supertagger performs much better on prepositions. Prepositions have different categories when appearing as arguments or adjuncts, and the distinction in the gold-standard was made using somewhat arbitrary heuristics (Hockenmaier and Steedman, 2007). It seems our embeddings have failed to capture these subtleties. Future work should explore methods for combining the strengths of each model.

Word Type	C&C Accuracy	Turian-50 Embeddings Accuracy
Verbs	93.9%	94.8%
Nouns	97.7%	97.3%
WH-words	90.1%	93.4%
Prepositions	94.8%	91.2%

Table 5: Supertagging accuracy on different types of words, with an ambiguity of 1.27 tags per word (corresponding to the C&C's initial beam setting). Overall performance with this beam is almost identical. Words types were identified using gold POS tags, using *IN* for prepositions.

4.7 Discussion

With a narrow supertagger beam, our method gives similar results to the C&C supertagger. However, it gains by being more accurate on difficult cases, due to not relying on lexical or POS features. These improvements lead directly to parser improvements. We identify two cases where our supertagger greatly outperforms the C&C parser: where the POS-tag is incorrect, and where the word-category pair is unseen in the labelled data. Our approach achieves larger improvements out-of-domain than in-domain, suggesting that the large vocabulary of embeddings built by the unsupervised pre-training allows it to better generalize from the labelled data.

Interestingly, the best-performing Turian-50 embeddings were trained on just 37M words of text (compared to 100B words for the Skip-gram embeddings), suggesting that further improvements may well be possible using larger unlabelled corpora. Future work should investigate whether the models and embeddings that work well for supertagging generalize to other tasks.

5 Related Work

Many methods have recently been proposed for improving supervised parsers with unlabelled data. Most of these are orthogonal to our work, and larger improvements may be possible by combining them.

Thomforde and Steedman (2011) extends a CCG lexicon by inferring categories for unseen words, based on the likely categories of surrounding words. Unlike our method, this approach is able to learn

categories which were unseen in the labelled data, which is shown to be useful for parsing a corpus of questions. Deoskar et al. (2011) and Deoskar et al. (2014) use Viterbi-EM to learn new lexical entries by running a generative parser over a large unlabelled corpus. They show good improvements in accuracy on unseen words, but not overall parsing improvements in-domain. Their parsing model aims to capture non-local information about word usage, which would not be possible for the local context windows used to learn our embeddings.

Self-training is another popular method for domain adaptation, and was used successfully by Honnibal et al. (2009) to improve CCG parser performance on Wikipedia. However, it caused a decrease in performance on the in-domain data, and our method achieves better performance across all domains. McClosky et al. (2006) improve a Penn Treebank parser in-domain using self-training, but other work has failed to improve performance out-of-domain using self training (Dredze et al., 2007). In a similar spirit to our work, Koo et al. (2008) improve parsing accuracy using unsupervised word cluster features—we have shown that word-embeddings outperform Brown clusters for CCG supertagging.

An alternative approach to domain adaptation is to annotate a small corpus of out-of-domain text. Rimell and Clark (2008) argue that this annotation is simpler for lexicalized formalisms such as CCG, as large improvements can be gained from annotating lexical categories, rather than full syntax trees. They achieve higher parsing accuracies than us on biomedical text, but our unsupervised method requires no annotation. It seems likely that our method could be further improved by incorporating out-of-domain labelled data (where available).

The best reported CCG parsing results have been achieved with a model that integrates supertagging and parsing (Auli and Lopez, 2011a; Auli and Lopez, 2011b). This work still uses the same feature set as the C&C parser, suggesting further improvements may be possible by using our embeddings features. Auli and Lopez POS-tag the sentence before parsing, but using our features would allow us to fully eliminate the current pipeline approach to CCG parsing.

Our work also builds on approaches to semi-supervised NLP using neural embeddings (Turian et al., 2010; Collobert et al., 2011b). Existing work has mainly focused on 'flat' tagging problems, without hierarchical structure. Collobert (2011) gives a model for parsing using embeddings features, by treating each level of the parse tree as a sequence classification problem. Socher et al. (2013) introduce a model in which context-free grammar parses are reranked based on compositional distributional representations for each node. Andreas and Klein (2014) experiment with a number of approaches to improving the Berkeley parser with word embeddings. Such work has not improved over state-of-the-art existing feature sets for constituency parsing—although Bansal et al. (2014) achieve good results for dependency parsing using embeddings. CCG categories contain much of the hierarchical structure needed for parsing, giving a simpler way to improve a parser using embeddings.

6 Conclusions

We have shown that CCG parsing can be significantly improved by predicting lexical categories based on unsupervised word embeddings. The resulting parsing pipeline is simpler, and has improved performance both in and out of domain. We expect further improvements to follow as better word embeddings are developed, without other changes to our model. Our approach reduces the problem of sparsity caused by the large number of CCG categories, suggesting that finer-grained categories could be created for CCGBank (in the spirit of Honnibal et al. (2010)), which lead to improved performance in downstream semantic parsers. Future work should also explore domain-adaptation, either using unsupervised embeddings trained on out-of-domain text, or using supervised training on out-of-domain corpora. Our results also have implications for other NLP tasks—suggesting that using word embeddings features may be particularly useful out-of-domain, in pipelines that currently rely on POS taggers, and in tasks which suffer from sparsity in the labelled data.

Code for our supertagger is released as part of the EASYCCG parser (Lewis and Steedman, 2014), available from: `https://github.com/mikelewis0/easyccg`

Acknowledgments

We would like to thank Tejaswini Deoskar, Bharat Ram Ambati, Michael Roth and the anonymous reviewers for helpful feedback on an earlier version of this paper. We also thank Rahul Jha for sharing his re-implementation of Collobert et al. (2011b)'s model, and Stephen Clark, Laura Rimell and Matthew Honnibal for making their out-of-domain CCG corpora available.

References

Jacob Andreas and Dan Klein. 2014. How much do word embeddings encode about syntax. In *Proceedings of ACL*.

Michael Auli and Adam Lopez. 2011a. A comparison of loopy belief propagation and dual decomposition for integrated CCG supertagging and parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 470–480. Association for Computational Linguistics.

Michael Auli and Adam Lopez. 2011b. Training a log-linear parser with loss functions via softmax-margin. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 333–343. Association for Computational Linguistics.

Srinivas Bangalore and Aravind K Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Johan Bos. 2008. Wide-coverage semantic analysis with boxer. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, Research in Computational Semantics, pages 277–286. College Publications.

P.F. Brown, P.V. Desouza, R.L. Mercer, V.J.D. Pietra, and J.C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Stephen Clark and James R Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. 2011a. Torch7: A Matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011b. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudk, editors, *AISTATS*, volume 15 of *JMLR Proceedings*, pages 224–232. JMLR.org.

James R Curran, Stephen Clark, and David Vadas. 2006. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 697–704. Association for Computational Linguistics.

Tejaswini Deoskar, Markos Mylonakis, and Khalil Sima'an. 2011. Learning structural dependencies of words in the zipfian tail. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 80–91. Association for Computational Linguistics.

Tejaswini Deoskar, Christos Christodoulopoulos, Alexandra Birch, and Mark Steedman. 2014. Generalizing a Strongly Lexicalized Parser using Unlabeled Data. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, Joao Graca, and Fernando Pereira. 2007. Frustratingly hard domain adaptation for dependency parsing. In *EMNLP-CoNLL*, pages 1051–1055.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Matthew Honnibal, Joel Nothman, and James R Curran. 2009. Evaluating a statistical CCG parser on Wikipedia. In *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pages 38–41. Association for Computational Linguistics.

M. Honnibal, J.R. Curran, and J. Bos. 2010. Rebanking CCGbank for improved NP interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 207–215. Association for Computational Linguistics.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing.

Jayant Krishnamurthy and Tom M. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 754–765. Association for Computational Linguistics.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1223–1233. Association for Computational Linguistics.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556, Seattle, Washington, USA, October. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2013a. Combined Distributional and Logical Semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Mike Lewis and Mark Steedman. 2013b. Unsupervised induction of cross-lingual semantic relations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 681–692, Seattle, Washington, USA, October. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2014. A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, October.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomáš Mikolov. 2012. *Statistical language models based on neural networks*. Ph.D. thesis, Ph. D. thesis, Brno University of Technology.

Andriy Mnih and Geoffrey E Hinton. 2008. A scalable hierarchical distributed language model. In *NIPS*, pages 1081–1088.

Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gómez Rodríguez. 2010. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 833–841, Beijing, China, August. Coling 2010 Organizing Committee.

Naoaki Okazaki. CRFsuite: a fast implementation of conditional random fields (CRFs).

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. 2007. Bioinfer: a corpus for information extraction in the biomedical domain. *BMC bioinformatics*, 8(1):50.

Laura Rimell and Stephen Clark. 2008. Adapting a lexicalized-grammar parser to contrasting domains. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 475–484. Association for Computational Linguistics.

Laura Rimell and Stephen Clark. 2009. Porting a lexicalized-grammar parser to the biomedical domain. *Journal of Biomedical Informatics*, 42(5):852–865.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 813–821. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

Emily Thomforde and Mark Steedman. 2011. Semi-supervised CCG lexicon extension. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1246–1256. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.