How much do word embeddings encode about syntax?

Jacob Andreas and **Dan Klein**
Computer Science Division
University of California, Berkeley
{jda,klein}@cs.berkeley.edu

Abstract

Do continuous word embeddings encode any useful information for constituency parsing? We isolate three ways in which word embeddings might augment a state-of-the-art statistical parser: by connecting out-of-vocabulary words to known ones, by encouraging common behavior among related in-vocabulary words, and by directly providing features for the lexicon. We test each of these hypotheses with a targeted change to a state-of-the-art baseline. Despite small gains on extremely small supervised training sets, we find that extra information from embeddings appears to make little or no difference to a parser with adequate training data. Our results support an overall hypothesis that word embeddings import syntactic information that is ultimately redundant with distinctions learned from treebanks in other ways.

1 Introduction

This paper investigates a variety of ways in which word embeddings might augment a constituency parser with a discrete state space. Word embeddings—representations of lexical items as points in a real vector space—have a long history in natural language processing, going back at least as far as work on latent semantic analysis (LSA) for information retrieval (Deerwester et al., 1990). While word embeddings can be constructed directly from surface distributional statistics, as in LSA, more sophisticated tools for unsupervised extraction of word representations have recently gained popularity (Collobert et al., 2011; Mikolov et al., 2013a). Semi-supervised and unsupervised models for a variety of core NLP tasks, including named-entity recognition (Freitag, 2004), part-of-speech tagging (Schütze, 1995), and chunking (Turian et al., 2010) have been shown to benefit from the inclusion of word embeddings as features. In the other direction, access to a syntactic parse has been shown to be useful for constructing word embeddings for phrases compositionally (Hermann and Blunsom, 2013; Andreas and Ghahramani, 2013). *Dependency* parsers have seen gains from distributional statistics in the form of discrete word clusters (Koo et al., 2008), and recent work (Bansal et al., 2014) suggests that similar gains can be derived from embeddings like the ones used in this paper.

It has been less clear how (and indeed whether) word embeddings in and of themselves are useful for *constituency* parsing. There certainly exist competitive parsers that internally represent lexical items as real-valued vectors, such as the neural network-based parser of Henderson (2004), and even parsers which use pre-trained word embeddings to represent the lexicon, such as Socher et al. (2013). In these parsers, however, use of word vectors is a structural choice, rather than an added feature, and it is difficult to disentangle whether vector-space lexicons are actually more powerful than their discrete analogs—perhaps the performance of neural network parsers comes entirely from the model's extra-lexical syntactic structure. In order to isolate the contribution from word embeddings, it is useful to demonstrate improvement over a parser that already achieves state-of-the-art performance *without* vector representations.

The fundamental question we want to explore is whether embeddings provide any information beyond what a conventional parser is able to induce from labeled parse trees. It could be that the distinctions between lexical items that embeddings capture are already modeled by parsers in other ways and therefore provide no further benefit. In this paper, we investigate this question empirically, by isolating three potential mechanisms for improvement from pre-trained word embeddings. Our result is mostly negative. With extremely lim-

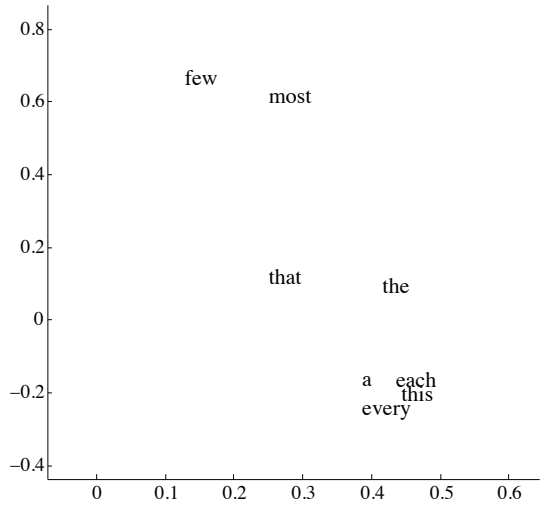

Figure 1: Word representations of English determiners, projected onto their first two principal components. Embeddings from Collobert et al. (2011).

ited training data, parser extensions using word embeddings give modest improvements in accuracy (relative error reduction on the order of 1.5%). However, with reasonably-sized training corpora, performance does not improve even when a wide variety of embedding methods, parser modifications, and parameter settings are considered.

The fact that word embedding features result in nontrivial gains for discriminative dependency parsing (Bansal et al., 2014), but do not appear to be effective for constituency parsing, points to an interesting structural difference between the two tasks. We hypothesize that dependency parsers benefit from the introduction of features (like clusters and embeddings) that provide syntactic abstractions; but that constituency parsers already have access to such abstractions in the form of supervised preterminal tags.

2 Three possible benefits of word embeddings

We are interested in the question of whether a state-of-the-art discrete-variable constituency parser can be improved with word embeddings, and, more precisely, what aspect (or aspects) of the parser can be altered to make effective use of embeddings.

It seems clear that word embeddings exhibit some syntactic structure. Consider Figure 1, which shows embeddings for a variety of English determiners, projected onto their first two principal components. We can see that the quantifiers *each* and *every* cluster together, as do *few* and *most*. These are precisely the kinds of distinctions between determiners that state-splitting in the Berkeley parser has shown to be useful (Petrov and Klein, 2007), and existing work (Mikolov et al., 2013b) has observed that such regular embedding structure extends to many other parts of speech. But we don't know how prevalent or important such "syntactic axes" are in practice. Thus we have two questions: Are such groupings (learned on large data sets but from less syntactically rich models) better than the ones the parser finds on its own? How much data is needed to learn them without word embeddings?

We consider three general hypotheses about how embeddings might interact with a parser:

1. **Vocabulary expansion hypothesis**: Word embeddings are useful for handling *out-of-vocabulary* words, because they automatically ensure that unknown words are treated the same way as known words with similar representations. Example: the infrequently-occurring treebank tag UH dominates greetings (among other interjections). Upon encountering the unknown word *hey*, the parser assigns a low posterior probability of having been generated from UH. But its distributional representation is very close to the known word *hello*, and a model capable of mapping *hey* to its neighbor should be able to assign the right tag.

2. **Statistic sharing hypothesis**: Word embeddings are useful for handling *in-vocabulary* words, by making it possible to pool statistics for related words. Example: individual first names are also rare in the treebank, but tend to cluster together in distributional representations. A parser which exploited this effect could use this to acquire a robust model of name behavior by sharing statistics from all first names together, preventing low counts from producing noisy models of names.

3. **Embedding structure hypothesis**: The structure of the space used for the embeddings directly encodes syntactic information in its coordinate axes. Example: with the exception of *a*, the vertical axis in Figure 1 seems to group words by definiteness. We would expect a feature corresponding to a word's position along this axis to be a useful feature in a feature-based lexicon.

Note that these hypotheses are not all mutually exclusive, and two or all of them might provide independent gains. Our first task is thus to design a set of orthogonal experiments which make it possible to test each of the three hypotheses in isolation. It is also possible that other mechanisms are at play that are not covered by these three hypotheses, but we consider these three to be likely central effects.

3 Parser extensions

For the experiments in this paper, we will use the Berkeley parser (Petrov and Klein, 2007) and the related Maryland parser (Huang and Harper, 2011). The Berkeley parser induces a latent, state-split PCFG in which each symbol V of the (observed) X-bar grammar is refined into a set of more specific symbols $\{V_1, V_2, \ldots\}$ which capture more detailed grammatical behavior. This allows the parser to distinguish between words which share the same tag but exhibit very different syntactic behavior—for example, between articles and demonstrative pronouns. The Maryland parser builds on the state-splitting parser, replacing its basic word emission model with a feature-rich, log-linear representation of the lexicon.

The choice of this parser family has two motivations. First, these parsers are among the best in the literature, with a test performance of 90.7 F_1 for the baseline Berkeley parser on the Wall Street Journal corpus (compared to 90.4 for Socher et al. (2013) and 90.1 for Henderson (2004)). Second, and more importantly, the fact that they use no continuous state representations internally makes it easy to design experiments that isolate the contributions of word vectors, without worrying about effects from real-valued operators higher up in the model. We consider the following extensions:

Vocabulary expansion $\rightarrow$ OOV model
To evaluate the vocabulary expansion hypothesis, we introduce a simple but targeted out-of-vocabulary (OOV) model in which every unknown word is simply replaced by its nearest neighbor in the training set. For OOV words which are not in the dictionary of embeddings, we back off to the unknown word model for the underlying parser.

Statistic sharing $\rightarrow$ Lexicon pooling model
To evaluate the statistic sharing hypothesis, we propose a novel smoothing technique. The Berkeley lexicon stores, for each latent (tag, word) pair, the probability $p(w|t)$ directly in a lookup table. If we want to encourage similarly-embedded words to exhibit similar behavior in the generative model, we need to ensure that the are preferentially mapped onto the same latent preterminal tag. In order to do this, we replace this direct lookup with a smoothed, kernelized lexicon, where:

$$p(w|t) = \frac{1}{Z} \sum_{w'} \alpha_{t,w'} e^{-\beta ||\phi(w) - \phi(w')||^2} \qquad (1)$$

with Z a normalizing constant to ensure that $p(\cdot|t)$ sums to one over the entire vocabulary. $\phi(w)$ is the vector representation of the word w, $\alpha_{t,w}$ are per-basis weights, and β is an inverse radius parameter which determines the strength of the smoothing. Each $\alpha_{t,w}$ is learned in the same way as its corresponding probability in the original parser model—during each M step of the training procedure, $\alpha_{w,t}$ is set to the expected number of times the word w appears under the refined tag t. Intuitively, as β grows small groups of related words will be assigned increasingly similar probabilities of being generated from the same tag (in the limit where $\beta = 0$, Equation 1 is a uniform distribution over the entire vocabulary). As β grows large words become more independent (and in the limit where $\beta = \infty$, each summand in Equation 1 is zero except where $w' = w$, and we recover the original direct-lookup model).

There are computational concerns associated with this approach: the original scoring procedure for a (word, tag) pair was a single (constant-time) lookup; here it might take time linear in the size of the vocabulary. This causes parsing to become unacceptably slow, so an approximation is necessary. Luckily, the exponential decay of the kernel ensures that each word shares most of its weight with a small number of close neighbors, and almost none with words farther away. To exploit this, we pre-compute the k-nearest-neighbor graph of points in the embedding space, and take the sum in Equation 1 only over this set of nearest neighbors. Empirically, taking $k = 20$ gives adequate performance, and increasing it does not seem to alter the behavior of the parser.

As in the OOV model, we also need to worry about how to handle words for which we have no vector representation. In these cases, we simply treat the words as if their vectors were so far away from everything else they had no influence, and report their weights as $p(w|t) = \alpha_w$. This ensures that our model continues to include the original Berkeley parser model as a limiting case.

Embedding structure → embedding features To evaluate the embedding structure hypothesis, we take the Maryland featured parser, and replace the set of lexical template features used by that parser with a set of indicator features on a discretized version of the embedding. For each dimension i, we create an indicator feature corresponding to the linearly-bucketed value of the feature at that index. In order to focus specifically on the effect of word embeddings, we remove the morphological features from the parser, but retain indicators on the identity of each lexical item.

The extensions we propose are certainly not the only way to target the hypotheses described above, but they have the advantage of being minimal and straightforwardly interpretable, and each can be reasonably expected to improve parser performance if its corresponding hypothesis is true.

4 Experimental setup

We use the Maryland implementation of the Berkeley parser as our baseline for the kernel-smoothed lexicon, and the Maryland featured parser as our baseline for the embedding-featured lexicon.[1] For all experiments, we use 50-dimensional word embeddings. Embeddings labeled C&W are from Collobert et al. (2011); embeddings labeled CBOW are from Mikolov et al. (2013a), trained with a context window of size 2.

Experiments are conducted on the Wall Street Journal portion of the English Penn Treebank. We prepare three training sets: the complete training set of 39,832 sentences from the treebank (sections 2 through 21), a smaller training set, consisting of the first 3000 sentences, and an even smaller set of the first 300.

Per-corpus-size settings of the parameter β are set by searching over several possible settings on the development set. For each training corpus size we also choose a different setting of the number of splitting iterations over which the Berkeley parser is run; for 300 sentences this is two splits, and for 3000 four splits. This is necessary to avoid overfitting on smaller training sets. Consistent with the existing literature, we stop at six splits when using the full training corpus.

Model		300	3000	Full
Baseline		71.88	84.70	91.13
OOV	(C&W)	72.20	84.77	91.22
OOV	(CBOW)	72.20	84.78	91.22
Pooling	(C&W)	72.21	84.55	91.11
Pooling	(CBOW)	71.61	84.73	91.15
Features	(ident)	67.27	82.77	90.65
Features	(C&W)	70.32	83.78	91.08
Features	(CBOW)	69.87	84.46	90.86

Table 1: Contributions from OOV, lexical pooling and featured models, for two kinds of embeddings (C&W and CBOW). For both choices of embedding, the pooling and OOV models provide small gains with very little training data, but no gains on the full training set. The featured model never achieves scores higher than the generative baseline.

Model	300	3000	Full
Baseline	72.02	84.09	90.70
Pool + OOV (C&W)	72.43*	84.36*	90.11

Table 2: Test set experiments with the best combination of models (based on development experiments). Again, we observe small gains with restricted training sets but no gains on the full training set. Entries marked * are statistically significant ($p < 0.05$) under a paired bootstrap resampling test.

5 Results

Various model-specific experiments are shown in Table 1. We begin by investigating the OOV model. As can be seen, this model alone achieves small gains over the baseline for a 300-word training corpus, but these gains become statistically insignificant with more training data. This behavior is almost completely insensitive to the choice of embedding.

Next we consider the lexicon pooling model. We began by searching over exponentially-spaced values of β to determine an optimal setting for each training set size; as expected, for small settings of β (corresponding to aggressive smoothing) performance decreased; as we increased the parameter, performance increased slightly before tapering off

[1] Both downloaded from `https://code.google.com/p/umd-featured-parser/`

Experiment	WSJ → Brown	French
Baseline	86.36	74.84
Pool + OOV	86.42	75.18

Table 3: Experiments for other corpora, using the same combined model (lexicon pooling and OOV) as in Table 2. Again, we observe no significant gains over the baseline.

to baseline parser performance. The first block in Table 1 shows the best settings of β for each corpus size; as can be seen, this also gives a small improvement on the 300-sentence training corpus, but no discernible once the system has access to a few thousand labeled sentences.

Last we consider a model with a featured lexicon, as described in Huang and Harper (2011). A baseline featured model ("ident") contains only indicator features on word identity (and performs considerably worse than its generative counterpart on small data sets). As described above, the full featured model adds indicator features on the bucketed value of each dimension of the word embedding. Here, the trend observed in the other two models is even more prominent—embedding features lead to improvements over the featured baseline, but in no case outperform the standard baseline with a generative lexicon.

We take the best-performing combination of all of these models (based on development experiments, a combination of the lexical pooling model with $\beta = 0.3$, and OOV, both using C&W word embeddings), and evaluate this on the WSJ test set (Table 2). We observe very small (but statistically significant) gains with 300 and 3000 train sentences, but a decrease in performance on the full corpus.

To investigate the possibility that improvements from embeddings are exceptionally difficult to achieve on the Wall Street Journal corpus, or on English generally, we perform (1) a domain adaptation experiment, in which we use the OOV and lexicon pooling models to train on WSJ and test on the first 4000 sentences of the Brown corpus (the "WSJ → Brown" column in Table 3), and (2) a multilingual experiment, in which we train and test on the French treebank (the "French" column). Apparent gains from the OOV and lexicon pooling models remain so small as to be statistically indistinguishable.

6 Conclusion

With the goal of exploring how much useful syntactic information is provided by unsupervised word embeddings, we have presented three variations on a state-of-the-art parsing model, with extensions to the out-of-vocabulary model, lexicon, and feature set. Evaluation of these modified parsers revealed modest gains on extremely small training sets, which quickly vanish as training set size increases. Thus, at least restricted to phenomena which can be explained by the experiments described here, our results are consistent with two claims: (1) unsupervised word embeddings do contain some syntactically useful information, but (2) this information is redundant with what the model is able to determine for itself from only a small amount of labeled training data.

It is important to emphasize that these results do not argue against the use of continuous representations in a parser's state space, nor argue more generally that constituency parsers cannot possibly benefit from word embeddings. However, the failure to uncover gains when searching across a variety of possible mechanisms for improvement, training procedures for embeddings, hyperparameter settings, tasks, and resource scenarios suggests that these gains (if they do exist) are *extremely* sensitive to these training conditions, and not nearly as accessible as they seem to be in dependency parsers. Indeed, our results suggest a hypothesis that word embeddings are useful for dependency parsing (and perhaps other tasks) because they provide a level of syntactic abstraction which is explicitly annotated in constituency parses. We leave explicit investigation of this hypothesis for future work.

Acknowledgments

This work was partially supported by BBN under DARPA contract HR0011-12-C-0014. The first author is supported by a National Science Foundation Graduate Research Fellowship.

References

Jacob Andreas and Zoubin Ghahramani. 2013. A generative model of vector space semantics. In *Proceedings of the ACL Workshop on Continuous Vector Space Models and their Compositionality*, Sofia, Bulgaria.

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*.

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

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Dayne Freitag. 2004. Trained named entity recognition using distributional clusters. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 95. Association for Computational Linguistics.

Karl Moritz Hermann and Phil Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 894–904, Sofia, Bulgaria, August.

Zhongqiang Huang and Mary P. Harper. 2011. Feature-rich log-linear lexical model for latent variable pcfg grammars. In *Proceedings of the International Joint Conference on Natural Language Processing*, pages 219–227.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 595–603.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 746–751.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. Assocation for Computational Linguistics.

Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *Proceedings of the European Association for Computational Linguistics*, pages 141–148.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the Annual Meeting of the 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. Proceedings of the Annual Meeting of the Association for Computational Linguistics.