Prepositional Phrase Attachment over Word Embedding Products

Pranava Swaroop Madhyastha[†][*] **Xavier Carreras**[‡] **Ariadna Quattoni**[‡]
[†]University of Sheffield
p.madhyastha@sheffield.ac.uk
[‡]Naver Labs Europe
{xavier.carreras,ariadna.quattoni}@naverlabs.com

Abstract

We present a low-rank multi-linear model for the task of solving prepositional phrase attachment ambiguity (PP task). Our model exploits tensor products of word embeddings, capturing all possible conjunctions of latent embeddings. Our results on a wide range of datasets and task settings show that tensor products are the best compositional operation and that a relatively simple multi-linear model that uses only word embeddings of lexical features can outperform more complex non-linear architectures that exploit the same information. Our proposed model gives the current best reported performance on an out-of-domain evaluation and performs competively on out-of-domain dependency parsing datasets.

1 Introduction

The Prepositional Phrase (PP) attachment problem (Ratnaparkhi et al., 1994) is a classic ambiguity problem and is one of the main sources of errors for syntactic parsers (Kummerfeld et al., 2012).

Consider the examples in Figure 1. For the first case, the correct attachment is the prepositional phrase attaching to the *restaurant*, the noun. Whereas, in the second case the attachment site is the verb *went*. While the attachments are ambiguous, the ambiguity is more severe when unseen or infrequent words like *Hudson* are encountered.

Classical approaches for the task exploit a wide range of lexical, syntactic, and semantic features and make use of knowledge resources like WordNet and VerbNet (Stetina and Nagao, 1997; Agirre et al., 2008; Zhao and Lin, 2004).

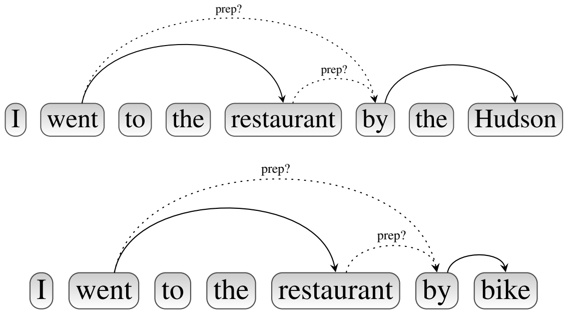


Figure 1: PP Attachment Ambiguity

In recent years, word embeddings have become a very popular representation for lexical items (Mikolov et al., 2013; Pennington et al., 2014). The idea is that the dimensions of a word embedding capture lexical, syntactic, and semantic features of words –in essence, the type of information that is exploited in PP attachment systems. Recent work in dependency parsing (Chen and Manning, 2014; Lei et al., 2014) suggests that these embeddings can also be useful to resolve PP attachment ambiguities. We follow this last line of research and investigate the use of word embeddings for PP attachment. Different from previous works, we consider several types of compositions for the vector embeddings corresponding to the words involved in a PP attachment decision. In particular, our model will define parameters over the tensor product of these embeddings. We control the capacity of the model by imposing low-rank constraints on the corresponding tensor which we formulate as a convex loss minimization.

We conduct experiments on several datasets and settings and show that this relatively simple multi-linear model can give performances comparable (and in some cases, even superior) than more complex neural network models that use the same information. Our results suggest that for the

[*] This work was carried out when the author was a PhD student at the Universitat Politècnica de Catalunya

Proceedings of the 15th International Conference on Parsing Technologies, pages 32–43,
Pisa, Italy; September 20–22, 2017. ©2017 Association for Computational Linguistics

PP attachment problem, exploring product spaces of dense word representations produces improvements in performance comparable to those obtained by incorporating non-linearities via a neural network.

Our main contributions are: a) we present a simple multi-linear model that makes use of tensor products of word embeddings, capturing all possible conjunctions of latent embeddings; b) we conduct comprehensive experiments of different embeddings and composition operations for PP attachment and observe that syntax infused embeddings perform significantly better; c) our proposed simple multi-linear model that uses only word embeddings can outperform complex non-linear architectures that exploit similar information; d) for out-of-domain evaluation sets, we observe significant improvements by using word embeddings trained from the source and target domains. With these imrpovements, our tensor products outperform state-of-the art dependency parsers on PP attachment decisions.

2 PP Attachment

Ratnaparkhi et al. (1994) first proposed a formulation of PP attachment as a binary prediction problem. The task is as follows: we are given a four-way tuple $\langle v, o, p, m \rangle$ where v is a verb, o is a noun object, p is a preposition, and m is a modifier noun; the goal is to decide whether the prepositional phrase $\langle p, m \rangle$ attaches to the verb v or to the noun object o.

More recently, Belinkov et al. (2014) proposed a generalization of PP attachment that considers multiple attachment candidates. Formally, we are given a tuple $\langle H, p, m \rangle$, where H is a set of candidate attachment tokens, and the goal is to decide what is the correct attachment for the $\langle p, m \rangle$ prepositional phrase. The binary case corresponds to $H = \{v, o\}$.

In this paper we use the generalized definition. Given a tuple $\langle H, p, m \rangle$, the models we present in this paper compute the following prediction:

$$\operatorname*{argmax}_{h \in H} f(h, p, m) \qquad , \qquad (1)$$

where f is a function that scores a candidate attachment h for the $\langle p, m \rangle$ phrase. Next section discusses several definitions of f based on tensor products of word embeddings.

3 Tensor Products for PP Attachment

For any word x in the vocabulary, we denote as $\boldsymbol{v}_x \in \mathcal{R}^n$ the n-dimensional vector for w, known as the word embedding of w. We will assume access to existing word embeddings for all words in our data.

Let $\boldsymbol{a} \in \mathcal{R}^{n_1}$ and $\boldsymbol{b} \in \mathcal{R}^{n_2}$ be two vectors. We denote as $\boldsymbol{a} \otimes \boldsymbol{b} \in \mathcal{R}^{n_1 * n_2}$ the Kronecker product of the two vectors, which results in a vector that has one dimension for any two dimensions of the argument vectors: the product of the i-th coordinate of $\boldsymbol{a}$ times the j-th coordinate of $\boldsymbol{b}$ results in the $(i-1) * n_1 + j$ coordinate of $\boldsymbol{a} \otimes \boldsymbol{b}$.

The tensor product model for PP attachment is as follows (see also Figure 2):

$$f(h, p, m) = \boldsymbol{v}_h^\top \, \boldsymbol{W} \, [\boldsymbol{v}_p \otimes \boldsymbol{v}_m] \qquad , \qquad (2)$$

where $\boldsymbol{W} \in \mathcal{R}^{n \times n^2}$ is a matrix of parameters, taking the embedding of the attachment candidate h on the left, and the product of embeddings of the $\langle p, m \rangle$ phrase on the right.

This is a multi-linear function: it is a function that is non-linear on each of the three argument vectors, but is linear in their product. Thus, our model is exploiting all conjunctions of *latent features* present in the word embeddings, resulting in a cubic number of parameters with respect to n. We note that if we pre-process the word embeddings to have a special dimension fixed to 1, then our model has parameters for each of the word embeddings alone, all binary conjunctions between any two vectors, and all ternary conjunctions.

Equation (2) is a multi-linear tensor written as a bilinear form. That is, we unfold the tensor into a matrix $\boldsymbol{W}$ that groups vectors based on the nature of the attachment problem: the vector for the head candidate is on the left side, while the vectors for the prepositional phrase are on the right side. Without any constraints on the parameters $\boldsymbol{W}$, this grouping is irrelevant. [1] However, our learning algorithm imposes low-rank constraints on $\boldsymbol{W}$ (see Section 3.2 below), for which the unfolding of the tensor becomes relevant.

3.1 Variations of the Tensor

We now discuss variations to the above model. In all cases we will write our models as bilinear func-

[1] In fact, we could choose to write a standard linear model between a weight vector and the Kronecker product of the three vectors: $\boldsymbol{w} \cdot [\boldsymbol{v}_h \otimes \boldsymbol{v}_p \otimes \boldsymbol{v}_m]$.

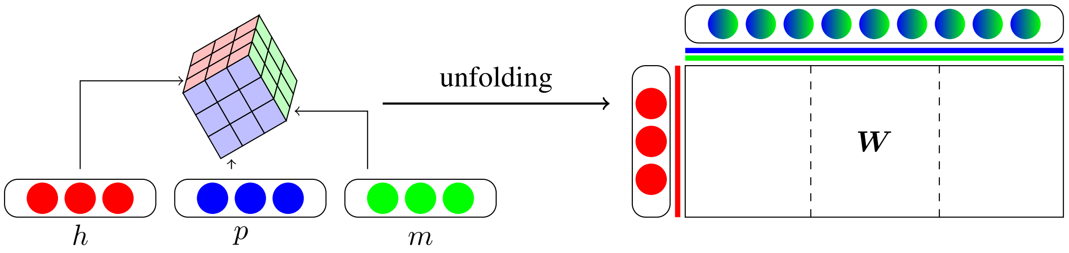


Figure 2: The tensor product of word embeddings. Here, h, p, and m are the head, preposition and modifier of the PP attachment structure, represented by their word embeddings. The tensor product forms a cube, which we unfold with respect to the head and the prepositional phrase. The resulting matrix $\boldsymbol{W} \in \mathcal{R}^{n \times n^2}$ has a row for each head dimension, and a column for each pair of preposition and modifier dimensions.

tions of the following form:

$$f(h,p,m) = \boldsymbol{\alpha}(h)^\top \boldsymbol{W} \boldsymbol{\beta}(p,m) \tag{3}$$

where $\boldsymbol{\alpha}$ is a representation vector of the attachment, and $\boldsymbol{\beta}$ is a representation vector of the prepositional phrase. Setting $\boldsymbol{\alpha}(h) = \boldsymbol{v}_h$ and $\boldsymbol{\beta}(p,m) = \boldsymbol{v}_p \otimes \boldsymbol{v}_m$ gives our basic tensor. These are the variations:

- **Sum and Concatenation:** Let us first consider variations of the prepositional phrase representation. Instead of using the product of embeddings, we can consider the sum $\boldsymbol{\beta}(p,m) = \boldsymbol{v}_p + \boldsymbol{v}_m$, or the concatenation $\boldsymbol{\beta}(p,m) = [\boldsymbol{v}_p; \boldsymbol{v}_m]$. These cases drastically reduce the expressivity and dimension of the $\boldsymbol{\beta}$ vector, from n^2 for the product to n for the sum, or $2n$ for the concatenation. Both sum, averaging and concatenation are common ways to compose word embeddings, while it is more rare to find compositions based on the product.

- **Preposition Identities:** Our basic model is defined essentially over word embeddings, and ignores the actual identity of the words in either sides. However, for PP attachment, it is common to have parameters for each preposition, and we can easily model this. Let $\mathcal{P}$ be the set of prepositions, and let $\boldsymbol{i}_p \in \mathcal{R}^{|\mathcal{P}|}$ be an indicator vector for preposition p. We can then set $\boldsymbol{\beta}(p,m) = \boldsymbol{i}_p \otimes \boldsymbol{v}_m$. Our model is now equivalent to writing:

  $$f(h,p,m) = \boldsymbol{v}_h^\top \boldsymbol{W}_p \boldsymbol{v}_m \tag{4}$$

  where we have one separate parameter matrix $\boldsymbol{W}_p \in \mathcal{R}^{n \times n}$ per preposition p. This is the model proposed by Madhyastha et al. (2014).

- **Positional Information:** Positional information often improves syntactic models in general, and PP attachment is no exception as shown by Belinkov et al. (2014). Following that work, we consider H to be *ordered* with respect to the distance of each candidate to the preposition, and we let δ_h be the position of element h (thus δ_h is 1 if h is the closest candidate to p, 2 if it's the 2nd closest, $\ldots$). In vector form, let $\boldsymbol{\delta}_h \in \mathcal{R}^{|H|}$ be a positional indicator vector for h (i.e. the coordinate δ_h is 1). We can now compose the word embedding of h with positional information as $\boldsymbol{\alpha}(h) = \boldsymbol{\delta}_h \otimes \boldsymbol{v}_h$, which is equivalent to writing:

  $$f(h,p,m) = \boldsymbol{v}_h^\top \boldsymbol{W}_{\delta_h} [\boldsymbol{v}_p \otimes \boldsymbol{v}_m] \quad . \tag{5}$$

  A neural network with a weight matrix for each position was proposed by Belinkov et al. (2014).

In the experimental section we present an empirical comparison of these variations, essentially showing that making tensor products of vector representations effectively results in more accurate attachment models.

3.2 Low-rank Matrix Learning

To learn the parameters we optimize the logistic loss with *nuclear norm regularization* (ℓ_*), an objective that favors matrices $\boldsymbol{W}$ that have low-rank (Srebro et al., 2004). This regularized objective has been used in previous work to learn low-rank matrices (Madhyastha et al., 2014), and has been shown to be very effective for feature spaces that are highly conjunctive (Primadhanty et al., 2015), such as those that result from tensor products of word embeddings.

In our basic model, the number of parameters is n^3 (where n is the size of the individual embeddings). If $\boldsymbol{W}$ has rank k, then we can rewrite $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{V}^\top$ where $\boldsymbol{U} \in \mathcal{R}^{n \times k}$ and $\boldsymbol{V} \in \mathcal{R}^{n^2 \times k}$. Thus the score function can we rewritten as a k-dimensional inner product between the left and right vectors projected down to k dimensions. If k is low, then the score is defined in terms of a few projected features, which can benefit generalization.

Specifically, let $\mathcal{T}$ be the training set. We optimize this convex objective:

$$\underset{\boldsymbol{W}}{\operatorname{argmin}} \operatorname{logistic}(\mathcal{T}, \boldsymbol{W}) + \lambda \|\boldsymbol{W}\|_\star \qquad (6)$$

which combines the logistic loss with the nuclear norm regularizer ($\|\boldsymbol{W}\|_\star$), weighted by the constant λ. To find the optimum, we follow previous work and use a simple optimization scheme based on Forward-Backward Splitting (FOBOS) (Duchi and Singer, 2009).

We describe FOBOS briefly in Algorithm 1. Essentially the algorithm works by first computing the gradient of the negative log likelihood function as can be observed in line 4: η_t is the step size, and $g(\boldsymbol{W}_t)$ is the gradient at time t (we choose $\eta_t = c/\sqrt{t}$). Now, for the proximal step. For ℓ_2 regularization, the weights are regularized by using geometric shrinkage (line 6). For nuclear norm regularization (ℓ_*), first Singular Value Decomposition (SVD) of the weight matrix is performed followed by iterative shrinkage and thresholding of the singular values (lines 8 and 9).

This algorithm has fast convergence rates, sufficient for our application. Many other optimization approaches are possible, for example one could express the regularizer as a convex constraint and utilize a projected gradient method which has a similar convergence rate. Proximal methods are slightly more simple to implement and we chose the proximal approach.

For nuclear norm based regularization, we are required to compute the Singular Value Decomposition of $\boldsymbol{W}$ at each iteration. In practice, for our experiments, the dimensions of $\boldsymbol{W}$ were relatively small, allowing fast SVD computations.

4 Experiments

This section presents experiments using tensor models for PP attachment. Our interest is to evaluate the accuracy of our models with respect to the type and size of word embeddings, and with respect to how these embeddings are composed. We start describing the data and word embeddings, and then present results on two settings, binary and multiple attachments, comparing to the state-of-the-art in each case.

Algorithm 1: FOBOS Algorithm

```
   Input: Gradient function $g$
   Constants : $\lambda$ (regularization factor), $T$ (max
               iterations) and $c$ (step size)
   Output: $\boldsymbol{W}_{t+1}$
1  $\boldsymbol{W}_1 = \boldsymbol{0}$
2  while $t < T$ do
3      $\eta_t = \frac{c}{\sqrt{t}}$
4      $\boldsymbol{W}_{t+0.5} = \boldsymbol{W}_t - \eta_t g(\boldsymbol{W}_t)$
5      if $\ell_2$ regularizer then
6          $\boldsymbol{W}_{t+1} = \frac{1}{1+\eta_t\lambda}\boldsymbol{W}_{t+0.5}$
7      else if $\ell_*$ regularizer then
8          $U\Sigma V^\top = \mathrm{SVD}(\boldsymbol{W}_{t+0.5})$
9          $\bar{\Sigma}_{i,i} = \max(\Sigma_{i,i} - \eta_t\lambda, 0)$
10         $\boldsymbol{W}_{t+1} = U\bar{\Sigma}V^\top$
11 end
```

4.1 Data and Evaluation

We use standard datasets for PP attachment for two settings: binary and multiple attachments. In both cases, the evaluation metric is the attachment accuracy. The details are as follows.

RRR Dataset. This is the classic English dataset for PP attachment proposed by Ratnaparkhi et al. (1994) (referred to as RRR dataset), which is extracted from the Penn TreeBank (PTB). The dataset contains 20,801 training samples of PP attachment tuples $\langle v, o, p, m \rangle$. We preprocess the data as in previous work (Collins and Brooks, 1999): we lowercase all tokens, map numbers to a special token `NUM` and symbols to `SYM`. We use the development set from PTB, with 4,039 samples, to compare various configurations of our model. For testing, we consider several test sets proposed in the literature: a) The test set from the RRR dataset, with 3,097 samples from the PTB. b) The New York Times test set (NYT) released by (Nakashole and Mitchell, 2015). It contains 293 test samples. c) Wikipedia test set (WIKI) by (Nakashole and Mitchell, 2015). It contains 381 test samples from Wikipedia. Because the texts are not news articles, this is an out-of-domain test.

Belinkov et al. (2014) Datasets. We use the datasets released by Belinkov et al. (2014) for

English and Arabic.[2] These datasets follow the generalized version of PP attachment, and each sample consists of a preposition p, the noun below the preposition m, and a list of possible attachment heads H (which contain candidate nouns and verbs in the same sentence of the prepositional phrase). The English dataset is extracted from PTB, and has 35,359 training samples and 1,951 test samples. The Arabic dataset is extracted from the SPMRL shared task data (Seddah et al., 2014), and consists of 40,121 training samples and 3,647 test samples.

4.2 Word Embeddings

As our models exploit pre-trained word embeddings, we perform experiments with a variety of types of word embeddings. We use two word embedding methods and estimate vectors using different data sources. The methods are: (a) Skip-gram (Mikolov et al., 2013): We use the Skip-gram model from `word2vec`, and induce embeddings of different dimensionalities: 50, 100 and 300. In all cases we use a window of size 5 during training.[3] (b) Skip-dep (Bansal et al., 2014): This is essentially a Skip-gram model that uses dependency trees to define the context words during training, thus it captures syntactic correlations. We trained 50, 100 and 300 dimensional dependency-based embeddings, using the setting described in Bansal et al. (2014) however we made use of TurboParser (Martins et al., 2013) to obtain dependency trees from the source data [4].

For evaluations on English, we use the following data sources to train word embeddings: (a) BLLIP (Charniak et al., 2000), with $\sim$1.8 million sentences and $\sim$43 million tokens of Wall Street Journal text (and excludes PTB evaluation sets); (b) English Wikipedia[5], with $\sim$13.1 million sentences and $\sim$129 million tokens; (c) The New York Times portion of the GigaWord corpus, with $\sim$52 million sentences and $\sim$1,253 million tokens.

For Arabic, we used pre-trained 100-dimensional word embeddings from the arTenTen corpus that are distributed with the data.

We created a special *unknown* vector for unseen words by averaging the word vectors of least frequent words (i.e., with frequency less than 5). Further, we appended a fixed dimension set to 1 to all word vectors. As explained in Section 3, when doing tensor compositions, this special dimension has the effect of keeping all lower-order conjunctions, including each elementary coefficient of the word embeddings and a bias term.

4.3 Experiments on the Binary Attachment Setting

This section presents a series of experiments using the classic binary setting by Ratnaparkhi et al. (1994).

Comparing Word Embeddings. We start comparing word embeddings of different types (Skip-gram and Skip-dep) trained on different source data, for different dimensions. For this comparison we use the tensor product model of Eq. 2, that resolves the attachment using only a product of word embeddings, and used ℓ_* regularization. Table 1 presents the results on the RRR development set. Looking at results using Skip-gram, we observe two clear trends that are expected: results improve whenever (1) we increase the dimensionality of the embeddings (n); and (2) we increase the size of the corpus used to induce the embeddings (BLLIP is the smallest, NYT is the largest).[6] When looking at the performance of models using Skip-dep vectors, which are induced using parse trees, then the results are better than when using Skip-gram. This is a signal that syntactic-based word embeddings favor PP attachment, which after all is a syntactic disambiguation task. We note that this was also found by Belinkov et al. (2014). The peak performance is for Skip-dep using 100 dimensional vectors trained on BLLIP.[7] For this test, we do not see a benefit from training on larger data.

Comparing Compositions. Our model composes word embeddings using tensor products. Section 3.1 presents variations that compose the prepositional phrase (i.e. the preposition and modifier vectors) in different ways. We now compare

[2] `http://groups.csail.mit.edu/rbg/code/pp`.

[3] In preliminary experiments we tried a window of 2, which performed worse in our setting. According to Bansal et al. (2014) with larger context window, words that are topically-related tend to get closer.

[4] `http://www.cs.cmu.edu/~ark/TurboParser`

[5] The corpus and preprocessing script were sourced from `http://mattmahoney.net/dc/textdata`.

[6] For this experimental comparison, we also tried Glove (Pennington et al., 2014), another popular word embedding method, but the results were generally inferior.

[7] Under the sign test, the difference between the best Skip-dep and Skip-gram models was significant with $p < 0.05$, but other differences between Skip-dep models were not.

Word Embedding		Accuracy wrt. dimension (n)		
Type	Source Data	$n = 50$	$n = 100$	$n = 300$
Skip-gram	BLLIP	83.23	83.77	83.84
Skip-gram	Wikipedia	83.74	84.25	84.22
Skip-gram	NYT	84.76	85.06	85.15
Skip-dep	BLLIP	85.52	**86.33**	85.97
Skip-dep	Wikipedia	84.23	84.39	84.32
Skip-dep	NYT	85.27	85.48	–
Skip-gram & Skip-dep	BLLIP	–	83.44	–

Table 1: Attachment accuracy on the RRR development set for tensor product models using different word embeddings. We vary the type of word embedding (Skip-gram, Skip-dep), the source data used to induce vectors (BLLIP, Wikipedia, NYT) and the dimensionality of the vectors (50, 100, 300). The last row "Skip-gram & Skip-dep" corresponds to the concatenation of two 50-dimensional word embeddings, for a total of 100 dimensions.

Composition of p and m		Tensor Size	Acc.
Sum	$[\boldsymbol{v}_p + \boldsymbol{v}_m]$	$n \times n$	84.42
Concatenation	$[\boldsymbol{v}_p; \boldsymbol{v}_m]$	$n \times 2n$	84.94
p Indicator	$[\boldsymbol{i}_p \otimes \boldsymbol{v}_m]$	$n \times \vert\mathcal{P}\vert * n$	84.36
Product	$[\boldsymbol{v}_p \otimes \boldsymbol{v}_m]$	$n \times n * n$	**85.52**

Table 2: Development accuracy for several ways of composing the word embeddings of the prepositional phrase. $\boldsymbol{i}_p \in \mathcal{R}^{|\mathcal{P}|}$ denotes an indicator vector for preposition p, where $\mathcal{P}$ is the set of prepositions.

these variants empirically, using Skip-dep vectors with $n = 50$ as word embeddings. Table 2 summarizes the accuracy results on the development set, where we compare: summing the two vectors; concatenating them; making the product of embeddings; or using indicator vectors for the preposition, which replicates the model by Madhyastha et al. (2014). The table also shows the size of the resulting tensor (we note that $|\mathcal{P}|$ is 66 for the RRR data, thus using a 50-dimensional embedding for p results in a more compact tensor than using p's identity). The results show that the product model is the best of all[8], despite the fact that the number of parameters is cubic in the dimension of the word embeddings. We observed the same trend for larger vectors.

Comparison to the State of the Art. We now present results on the test sets for the binary setting, and compare to the state-of-the-art. The results are in Table 3, which lists representative and top-performing methods of the literature, as well as our tensor product model running with three different word embeddings. Two of the representative systems we list are the back-off model by Collins and Brooks (1999), and the neural model by Belinkov et al. (2014), which composes word embeddings in a neural fashion. These two systems use no other information that the lexical items (i.e., explicit words or word embeddings). The other two systems, by Stetina and Nagao (1997) and Nakashole and Mitchell (2015), use additional features, and most notably semantic information from WordNet or other ontologies, which has been shown to be beneficial for PP attachment. In general, the results that our models obtain are remarkably good, despite the fact that we only combine word embeddings in a straightforward way. On the RRR test, with the exception of the classic result by Stetina and Nagao (1997), our method using Skip-dep embeddings clearly outperforms any other recent system. On the WIKI test our method is clearly the best, while on the NYT test, our system is behind that of Nakashole and Mitchell (2015) but it is still competitive. In terms of the embeddings we use, the table shows that for the RRR test, embeddings induced from BLLIP perform clearly better, while for the out-of-domain tests, the embeddings induced from NYT are slightly better for the Wikipedia test, and clearly better for the NYT test.

4.4 Experiments on the Multiple Attachment Setting

We now examine the performance of our models on the setting and data by Belinkov et al. (2014), which deals with multiple head candidates. We

[8] The differences, though, were not significant under the sign test.

Method	Word Embedding	Test Accuracy RRR	 WIKI	 NYT
Tensor product	Skip-gram, Wikipedia, $n = 100$	84.96	83.48	82.13
"	Skip-gram, NYT, $n = 100$	85.11	83.52	82.65
"	Skip-dep, BLLIP, $n = 100$	*86.13*	83.60	82.30
"	Skip-dep, Wikipedia, $n = 100$	85.01	83.53	82.10
"	Skip-dep, NYT, $n = 100$	85.49	**83.64**	*83.47*
Stetina and Nagao (1997) (*)		**88.1**	-	-
Collins and Brooks (1999)		84.1	72.7	80.9
Belinkov et al. (2014)		85.6	-	-
Nakashole and Mitchell (2015) (*)		84.3	79.3	**84.3**

Table 3: Accuracy results over the RRR, NYT and WIKI test sets. (*) indicates that the system uses additional semantic features.

perform experiments on both English and Arabic datasets. For this setting, following Belinkov et al. (2014), we found necessary to use positional information of the head candidate, as described by Eq. 5. Without it the performance was much worse (possibly because in this data, a large number of samples attach to the first or second candidate in the list —about 93% of cases on the English data).

Table 4 presents our results. For English, we present results for models trained with nuclear-norm (ℓ_*) and ℓ_2 regularization, using 50-dimensional embeddings. Imposing low-rank on the product tensor yields some gains with respect to ℓ_2, however the improvements are not drastic. This is probably because embeddings are already compressed representations, and even products of them do not result in overfitting to training. We obtain a slight gain by using 100-dimensional embeddings, which results in an accuracy of 88.4 for English and 81.1 for Arabic. In any case, one characteristic of low-rank regularization is the inherent compression of the tensor. Figure 3 plots accuracy versus rank for the tensor working with 50-dimensional embeddings composed with positional information[9]: with rank 50 the model obtains 88% of accuracy while reducing the number of parameters by a factor of 6. For PP attachment, this has a computational advantage: the prepositional phrase needs to be projected only once (from 2,601 dimensions to k, where k is the rank) for all the head candidates in the sentence.

We compare our method to a series of results by Belinkov et al. (2014). Their "basic" model uses Skip-gram, and like us, by moving to syntactic vectors (noted "syn") they observed a gain in accuracy. However, in this comparable setting, our model outperforms theirs by 1.3% in English and 2% in Arabic. They also explored adding standard features (from WordNet and VerbNet, noted "feat"), and combining everything (noted "full"), which then surpasses our results. Very recently, Yu et al. (2016) has used a tensor model that combines standard feature templates (again using WordNet) with word embeddings, with significant improvements; however they do not report results on combining word embeddings only, which is our focus.

	Test Accuracy Arabic	 English
Tensor product (n=50, ℓ_2)	-	87.8
Tensor product (n=50, ℓ_*)	-	88.3
Tensor product (n=100, ℓ_*)	*81.1*	*88.4*
Belinkov et al. (2014) (basic)	77.1	85.4
Belinkov et al. (2014) (syn)	79.1	87.1
Belinkov et al. (2014) (feat)	80.4	87.7
Belinkov et al. (2014) (full)	**82.6**	88.7
Yu et al. (2016)	-	**90.3**

Table 4: Test accuracy for PP attachment with multiple head candidates.

Comparison to Dependency Parsers. We now compare our tensor models to state-of-the-art dependency parsers, specifically looking at PP attachment decisions. For this comparison, we took the English Web Treebank (WTB) (Petrov and McDonald, 2012), which has annotated evaluation sets for five domains, and extracted PP-attachment tuples using the procedure described by Belinkov

[9] We consider 7 head positions, and word vectors are 51 dimensions in practicei (with the dummy dimention). Thus, the unfolded matrix $\boldsymbol{W}$ has 357 rows and 2,601 columns.

	PTB	Web Treebank Development						Web Treebank Test					
	Test	A	E	N	R	W	Avg	A	E	N	R	W	Avg
	(2523)	(814)	(1025)	(969)	(783)	(1064)	(4655)	(868)	(936)	(839)	(902)	(788)	(4333)
Tensor BLLIP	**89.0**	83.7	80.2	81.9	83.2	85.3	82.8	82.7	82.6	87.4	82.5	86.3	84.2
Tensor BLLIP+WTB	88.9	**86.2**	**81.8**	84.1	**83.7**	**86.7**	**84.5**	83.3	**85.2**	**90.1**	**85.9**	86.6	**86.1**
Stanford	87.3	80.3	79.7	**84.5**	81.5	84.9	82.3	79.3	79.7	85.7	82.2	83.8	82.0
Turbo 2nd	88.8	84.5	80.1	82.8	83.1	85.1	83.1	83.6	83.7	87.6	84.2	**87.8**	85.3
Turbo 3rd	88.9	85.1	80.4	83.3	83.3	84.8	83.3	**84.2**	84.5	87.6	84.4	87.6	85.6

Table 5: Comparison between tensor products and dependency parsers, on PP attachment tuples in the Penn Treebank test (PTB) and in the English Web Treebank (WTB) evaluation sets – with separate results for each domain: answers (A), emails (E), newsgroups (N), reviews (R), and weblogs (W). The number of evaluation instances in each set appears in parenthesis. The tensor products use embeddings trained on BLLIP and BLLIP+WTB, and for both $n = 100$.

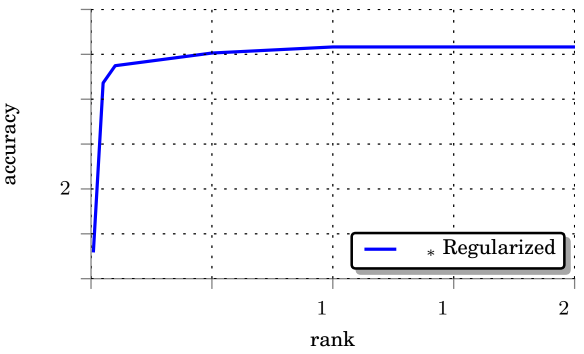


Figure 3: Accuracy versus rank of the tensor model on the English data by Belinkov et al. (2014). The tensor model uses 50-dimensional vectors composed with head position, and has size $357 \times 2,601$.

et al. (2014), resulting in 4,655 tuples on the development set and 4,333 tuples on the test set. We also applied the same procedure to the Penn Treebank test set, with 2.523 instances.[10] We selected two state-of-the-art dependency parsers which are publicly available. The first is the Stanford transition-based neural parser (Chen and Manning, 2014), which uses word embeddings but not as products.[11] The other is TurboParser (Martins et al., 2013)[12] which offers 2nd and 3rd order arc-factored models, with grandchildren features that capture the conjunction of the three words in a PP-attachment decision, even though those models do not use word embeddings. We ran the parsers on the evaluation sentences, and extracted the PP-attachment decision from the parse tree.[13] We also evaluated two 100-dimensional Skip-dep tensor products, one using embeddings trained on BLLIP, and a second one using embeddings trained on BLLIP and the unlabeled data from the Web Treebank.[14]

Table 5 presents the results. Comparing the tensor products, using PTB+WTB embeddings gives an improvement of 1.7% in accuracy on the WTB development test, for a slight decrease of 0.1% on the PTB test. This confirms that tensor products of word embeddings are a valid and simple approach to domain adaptation.

Comparing to parsers, our best tensor product performs better in almost all domains, and on average it performs significantly better in the WTB evaluation sets.[15] First, this confirms that PP attachment decisions are still an important source of errors of state-of-the-art parsers. And we see that a specialized model for PP attachment, despite its simplicity, can improve on these decisions.

Error Analysis. To further understand the performance of the tensor products and parsers on WTB development set, we consider PP attachment instances where the words are observed less than five times in the training data (1,565 cases out of

[10] The evaluation test by (Belinkov et al., 2014) has 1,951 instances. Hence, the results of our models are slightly different in this evaluation. We will release our extraction script.

[11] We used Stanford CoreNLP 3.7.0. We could not determine the characteristics of the embeddings in the model.

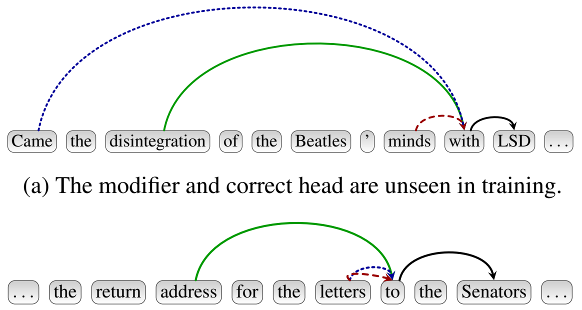
[12] We used version 2.3, available from `http://www.cs.cmu.edu/~ark/TurboParser`

[13] We ran all parsing models on correct PoS tags. Thus, these are optimistic performances. This choice rules out cases where the parsers fail because of tagging errors, which would be unfair because our models work on pre-selected head candidate lists which depend on correct PoS tags.

[14] We mixed the unlabeled data from all domains, for a total of ~4.7 million sentences and ~75.5 million tokens.

[15] Under the sign test, the differences on WTB evaluation sets between the tensor product on BLLIP+WTB and other models were significant: TurboParser with $p < 0.05$, the Stanford parser with $p < 0.01$, and the tensor product on BLLIP with $p < 0.01$.

4,655). The best tensor product obtains an accuracy of 84.3% (vs. 84.5%), while the 3rd order TurboParser gets 83.0% (vs. 83.3%) and the Stanford parser gets 81.3% (vs. 82.3%). The parsers suffer a drop, while the tensor model does not, suggesting that the tensor model is able to generalize better to less frequent words. Figure 4 shows two sample sentences from the Web Treebank that illustrate two cases of ambiguities. Sentence (a) is an example of lexical paucity, because the words of the attached phrase, *disintegration with LSD*, are absent in the training set. The tensor model correctly predicts the attachment, while the parsers do not. Sentence (b) is an example of sense ambiguity: the tensor model incorrectly predicts *address* as head of *to Senators*, which is plausible, but in this case the sentence is about the *return address* of the *letter to Senators*, which the parsers correctly predict. There are clear complementary benefits between parsers and products of embeddings, and these examples suggest combinations of both.



(a) The modifier and correct head are unseen in training.

(b) The correct head is ambiguous.

Figure 4: Examples from the Web Treebank development set, with the attachments predicted by the tensor product (solid green arc), the Stanford neural parser (dashed red arc) and the 3rd order TurboParser (dotted blue arc).

5 Related Work

5.1 Resolving PP Attachment Ambiguity

Several approaches have been proposed for solving the PP attachment problem, including maximum likelihood with back-off (Hindle and Rooth, 1993; Collins and Brooks, 1999), and discriminative training (Ratnaparkhi et al., 1994; Olteanu and Moldovan, 2005), among others. A key part of such systems is the representation they use, in the form of lexical, syntactic and semantic features of the main words involved in an attachment decision. Crucially, the best performing models are obtained when exploring conjunctions of such features. Some works have also explored using external knowledge resources in the form of ontologies and syntactic information (Stetina and Nagao, 1997; Zhao and Lin, 2004; Nakashole and Mitchell, 2015).

In our paper, we use word embeddings as the only source of lexical information. Previous work has explored word representations as extra features (Zhao and Lin, 2004). In our case, we define a model that exploits all conjunctions of the word vectors in an attachment decision. Our model is in fact a generalization of that of Madhyastha et al. (2014), as described in section 3.1. From that work, our application to PP attachment differs in using compact word embeddings as opposed to sparse distributional vectors. Mitchell and Lapata (2008) compared a variety of composition operations, including the tensor product, in the context of distributional lexical semantics.

Closely related to our work is the approach by Belinkov et al. (2014), who use neural networks that compose the embeddings of the words in the PP attachment structure. Their model composes word embeddings by first concatenating vectors and then projecting to a low-dimensional vector using a non-linear hidden layer. This basic composition block is used to define several compositional models for PP attachment. One difference is that we represent tensor products of embeddings, which result in projected hidden conjunctions when the tensor has low rank. In contrast, projecting concatenated embeddings results in hidden disjunctions of the input coefficients.

More recently, Yu et al. (2016) have also explored tensor models for PP attachment. Their focus is on representing standard feature templates (which are conjunctions of features of a variety of sources) as tensors, and on using low-rank constraints to favor parameter sharing among templates. One of their templates is the conjunction of the head, preposition and modifier (and word embeddings of these), which is the focus case of our paper. While there are differences in the way we learn a low-rank tensor (see below), they show superior performance, probably due to the combination of different features. Our experiments, in contrast, offer a controlled study over different aspects of word embeddings and their product.

Beyond applications to PP attachment, word embeddings have been used for a number of prediction tasks. In most cases, embeddings of two or more words are composed by concatenation – see (Turian et al., 2010; Chen and Manning, 2014; Dyer et al., 2015) to name a few, or averaging (Socher et al., 2011; Huang et al., 2012). Compositions based on product of embeddings have been explored in tensor models, which we discuss next.

5.2 Low Rank Tensors in NLP

Using tensors to represent products of elementary vectors has been a recent trend in NLP. Because most tasks in NLP benefit from exploiting conjunctions of elementary features, tensor models offer the appropriate framework for defining conjunctive feature spaces. A main benefit of the tensor representation is that it allows to control the model capacity using low-rank constraints. There are several ways to define the rank of a tensor, while for a matrix there is a unique definition. A natural and simple way to impose low-rank constraints on a tensor is by first unfolding the tensor into a matrix, and let the rank of the tensor be the rank the unfolded matrix. With this one can apply low-rank constraints by regularization, using the nuclear norm (which is a convex relaxation for low-rank regularization). In practice, this leads to a simple convex optimization that uses an SVD routine to solve the core part of the problem. This technique has been used recently for several problems (Balle and Mohri, 2012; Quattoni et al., 2014; Madhyastha et al., 2014, 2015; Primadhanty et al., 2015). There are 2^d ways to unfold a tensor of d modes. In our case, we have made the choice based on the application: we have grouped the preposition and modifier together. This choice has a clear computational advantage for the task: at prediction time, we can first project the prepositional phrase (which is fixed) to its low-dimensional representation, and then do the inner product with the projection of each head candidate. In general, one could try different unfoldings, or use multiple of them in a combination.

Another popular approach to low-rank tensor learning is directly optimizing over a low-rank decomposition of the tensor, such as the canonical polyadic or the Tucker forms (Lei et al., 2014; Fried et al., 2015; Yu et al., 2016). In the Tucker form, a tensor d modes has one projection matrix for each of the modes, where each projection matrix is a mapping from the original input vector space to a low-dimensional one, i.e. an embedding of the feature of the corresponding mode. One advantage of this approach is that there is no need to choose an unfolding. However, the optimization is non-convex.

6 Conclusion

We have described a simple PP attachment model based on tensor products of the word vectors in a PP attachment decision. We have established that the product of vectors improves over more simple compositions (based on sum or concatenation), while it remains computationally manageable due to the compact nature of word embeddings. In experiments on standard PP attachment datasets, our tensor models perform better than other methods using lexical information only, and are close in performance to methods using richer feature spaces. In out-of-domain tests we obtain improvements over state-of-the-art parsers. As our models only depend on word embeddings, this is a clear signal that word embeddings are appropriate representations to generalize to unseen structures.

By using low-rank constraints during learning we have observed small improvements over ℓ_2 regularization, but not drastic ones (compared to, for example, tensor compositions of sparse vectors, in which case low-rank constraints are generally much more beneficial). All in all, low-rank constraints are essential tools to control the capacity of tensor models. This framework is arguably more simple than neural compositions, because it avoids non-linearities and can be optimized with global routines like SVD. In our PP attachment experiments, we have obtained some gains in accuracy over the neural models by Belinkov et al. (2014) that use comparable representations. We have also obtain improvements over state-of-the-art dependency parsers.

In NLP, and in syntax in particular, there exist other paradigmatic lexical attachment ambiguities that, like PP attachment, can be framed within a particular scope of the dependency tree: adjectives, conjunctions, raising and control verbs, etc.. The tensor product we have presented can serve as a building block to define dependency parsing methods that make a central use of products of word embeddings.

References

Eneko Agirre, Timothy Baldwin, and David Martinez. 2008. Improving parsing and pp attachment performance with sense information. In *ACL*. pages 317–325.

Borja Balle and Mehryar Mohri. 2012. Spectral learning of general weighted automata via constrained matrix completion. *In Proceedings of Advances in Neural Information Processing Systems (NIPS)* .

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of Association of Computational Linguistics(ACL) Short Papers*.

Yonatan Belinkov, Tao Lei, Regina Barzilay, and Amir Globerson. 2014. Exploring compositional architectures and word vector representations for prepositional phrase attachment. *Transactions of the Association for Computational Linguistics* 2:561–572.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. BLLIP 1987-89 WSJ Corpus Release 1, LDC No. LDC2000T43. Linguistic Data Consortium.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 740–750. http://www.aclweb.org/anthology/D14-1082.

Michael Collins and James Brooks. 1999. Prepositional phrase attachment through a backed-off model. In *Natural Language Processing Using Very Large Corpora*, Springer, pages 177–189.

John Duchi and Yoram Singer. 2009. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research* 10:2899–2934.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 334–343. http://www.aclweb.org/anthology/P15-1033.

Daniel Fried, Tamara Polajnar, and Stephen Clark. 2015. Low-rank tensors for verbs in compositional distributional semantics. In *Short Papers of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*. pages 731–736.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational linguistics* 19(1):103–120.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 873–882.

Jonathan K Kummerfeld, David Hall, James R Curran, and Dan Klein. 2012. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 1048–1059.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 1381–1391. http://www.aclweb.org/anthology/P14-1130.

Pranava Swaroop Madhyastha, Xavier Carreras, and Ariadna Quattoni. 2014. Learning task-specific bilexical embeddings. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, pages 161–171. http://aclweb.org/anthology/C14-1017.

Pranava Swaroop Madhyastha, Xavier Carreras, and Ariadna Quattoni. 2015. Tailoring word embeddings for bilexical predictions: An experimental comparison. *In International Conference on Learning Representations (Workshop Contribution)* .

Andre Martins, Miguel Almeida, and A. Noah Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 617–622. http://aclweb.org/anthology/P13-2109.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *In Proceedings of International Conference on Learning Representations* .

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. pages 236–244.

Ndapandula Nakashole and Tom M Mitchell. 2015. A knowledge-intensive model for prepositional phrase attachment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics,(ACL)*. pages 365–375.

Marian Olteanu and Dan Moldovan. 2005. Pp-attachment disambiguation using large context. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 273–280.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL).

Audi Primadhanty, Xavier Carreras, and Ariadna Quattoni. 2015. Low-rank regularization for sparse conjunctive feature spaces: An application to named entity classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 126–135. http://www.aclweb.org/anthology/P15-1013.

Ariadna Quattoni, Borja Balle, Xavier Carreras, and Amir Globerson. 2014. Spectral regularization for max-margin sequence tagging. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. JMLR Workshop and Conference Proceedings, pages 1710–1718. http://jmlr.org/proceedings/papers/v32/quattoni14.pdf.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, pages 250–255.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the spmrl 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*. Dublin City University, Dublin, Ireland, pages 103–109. http://www.aclweb.org/anthology/W14-6111.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *In Proceedings of Advances in Neural Information Processing Systems*. pages 801–809.

Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. 2004. Maximum-margin matrix factorization. In *In proceedings of Advances in Neural Information Processing Systems (NIPS)*. pages 1329–1336.

Jiri Stetina and Makoto Nagao. 1997. Corpus based pp attachment ambiguity resolution with a semantic dictionary. In *Proceedings of the fifth workshop on very large corpora*.

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*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '10, pages 384–394. http://dl.acm.org/citation.cfm?id=1858681.1858721.

Mo Yu, Mark Dredze, Raman Arora, and Matthew R. Gormley. 2016. Embedding lexical features via low-rank tensors. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1019–1029. http://www.aclweb.org/anthology/N16-1117.

Shaojun Zhao and Dekang Lin. 2004. A nearest-neighbor method for resolving pp-attachment ambiguity. In *Natural Language Processing–IJCNLP 2004*, Springer, pages 545–554.