Universitat Politècnica de Catalunya
Departament de Ciències de la Computació
Programa de Doctorat en Intel·ligència Artificial

PhD Thesis

EXPLOITING WORD EMBEDDINGS FOR MODELING BILEXICAL RELATIONS

---

by Pranava Swaroop Madhyastha

Advised By:
Dr. Xavier Carreras
Dr. Ariadna Quattoni

May 2017

ABSTRACT

There has been an exponential surge of text data in the recent years. As a consequence, unsupervised methods that make use of this data have been steadily growing in the field of natural language processing (NLP). Word embeddings are low-dimensional vectors obtained using unsupervised techniques on the large unlabelled corpora, where words from the vocabulary are mapped to vectors of real numbers. Word embeddings aim to capture syntactic and semantic properties of words.

In NLP, many tasks involve computing the compatibility between lexical items under some linguistic relation. We call this type of relation a *bilexical relation*. Our thesis defines statistical models for bilexical relations that *centrally* make use of word embeddings. Our principle aim is that the word embeddings will favor generalization to words not seen during the training of the model.

The thesis is structured in four parts. In the first part of this thesis, we present a bilinear model over word embeddings that leverages a small supervised dataset for a binary linguistic relation. Our learning algorithm exploits low-rank bilinear forms and induces a low-dimensional embedding tailored for a target linguistic relation. This results in compressed task-specific embeddings.

In the second part of our thesis, we extend our bilinear model to a ternary setting and propose a framework for resolving prepositional phrase attachment ambiguity using word embeddings. Our models perform competitively with state-of-the-art models. In addition, our method obtains significant improvements on out-of-domain tests by simply using word-embeddings induced from source and target domains.

In the third part of this thesis, we further extend the bilinear models for expanding vocabulary in the context of statistical phrase-based machine translation. Our model obtains a probabilistic list of possible translations of target language words, given a word in the source language. We do this by projecting pre-trained embeddings into a common subspace using a log-bilinear model. We empirically notice a significant improvement on an out-of-domain test set.

In the final part of our thesis, we propose a non-linear model that maps initial word embeddings to task-tuned word embeddings, in the context of a neural network dependency parser. We demonstrate its use for improved dependency parsing, especially for sentences with unseen words. We also show downstream improvements on a sentiment analysis task.

ACKNOWLEDGMENTS

This thesis represents an amazing learning experience carefully guided by both my advisors Xavier Carreras and Ariadna Quattoni. I am most grateful and indebted to both my advisors for the patience and their encouragement throughout the process of the thesis. I have learned really a whole deal from the process of thinking, application of theory, implementation to writing from both my advisors. Their incredible knowledge and foresight helped me throughout the period of my PhD. I would also like to acknowledge the presence of Ziggy Lupus Pupusus who made our discussions a more fun and relaxing experience ;-) Thank you again Xavier and Ariadna, for all I know about doing research is all because of you!

I am very fortunate for having spent 6-months at TTIC with Mohit Bansal, Karen Livescu, and Kevin Gimpel. It was again an incredible journey and I learned how to focus on a problem from all three of you. TTIC also gave me the incredible opportunity of learning about research from different disciplines in computer science, the invited talks especially were a great motivational experience.

I also thank the anonymous reviewers of the thesis for their insightful comments and recommendations.

The work in this thesis has been made possible because of a great set of people around me. Lluís Padró for being a fantastic tutor. I always came to you in a hurry and you always resolved my problem in seconds. Solmaz Bagherpour, it was great talking to you during the time when we were the only two people in the lab, it was fun, also thanks for the Spanish help. Audi Primadhanty, for that period of time when we collaborated and hated the MUC dataset with a passion. Cristina España for being an awesome collaborator and mentor. Thanks, Pere Comas, Meritxell González, Xavier Lluís, Maria Fuentes, Alberto Barrón, and Roberto. I was always motivated by all your presence and the chats that we had during lunch. I am grateful to Marta Ruiz and José Fonollosa for those interesting collaborations. I also thank Arnau Ramisa, Edgar Simo-Serra and Francesc Moreno-Noguer for the collaborations. I especially am very greatful to Francesc for funding the unfortunate NAACL trip gaffe. Special mention to Daniele Pighin, it was incredible to chat with you when I was new at UPC.

I am thankful for the support I got at TTIC. Especially, John Wieting, Rasool Fakoor, Hermann Kamper, Hao Tang, Behnam Neyshabur and Dhivya for making it a great experience. I am also grateful for the opportunity at Xerox Research Center Europe, Grenoble, it was

really a stimulating experience to have met a lot of incredibly knowledgeable people. I especially enjoyed the lunch time conversations.

Finally, I am grateful to the support of my family and friends for being the stress busters. Specially my father and mother! Their dedication and help has made all this possible. Also, super special mention to Dasa!

This thesis would not have been possible without the necessary funding for my studies. The research in this thesis in many ways was funded directly by: FPI-UPC grant by UPC and Google Faculty Research Award to the TTIC collaborators which supported my stay. Several research projects contributed indirectly through my collaborators including, XLike (FP7- 288342), ERA-Net CHISTERA VISEN PCIN-2013-047, TACARDI (TIN2012-38523-C02-00), SKATER project (TIN2012-38584-C06-01), MINECO project RobInstruct TIN2014-58178-R and I-DRESS PCIN-2015-147.

CONTENTS

1 INTRODUCTION

1.1 CONTEXT

There has been an abundance of unstructured data generated over the web as text and other multimedia content. Most of these unstructured data contains information that could potentially help standard Natural Language Processing tasks (NLP). Unfortunately, it is often very difficult to analyze unstructured data and search for statistical cues that can be directly used for NLP models.

On the other end, state-of-the-art NLP models are usually supervised systems that are dependent on annotated data and rely on engineered features. The annotated data is often constrained over a single domain or a limited set of domains. NLP systems are increasingly being applied to domains such as web-based data, personal communications that include email, tweets, among many other domains. Designing features that result in a better overall performance of the system is usually an expensive task. One of the most important features commonly used in traditional NLP tasks is lexical features. However, the lexical statistics are often sparse as a significant portion of lexical entities is not observed in training data, limiting the generalization capabilities of the model.

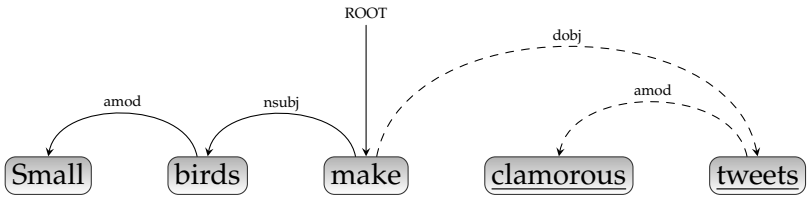


Figure 1.1: A sample sentence with *unseen* lexical items.

Let us consider a task of using a simple dependency parser that uses lexical information for the example sentence in Figure 1.1. To predict the correct dependency tree structure of the sentence, the parsing model would have to make several decisions using the lexical information. Now, let us assume that both *clamorous* and *tweets* are not seen in the corpus used to train the parser. It would have to accurately predict the dependency relation between both unseen words and their connection with other words in the sentence. However, in the absence of the correct lexical features the parser does not have enough information. These poor features result in uncertainty dur-

ing prediction of the correct dependency parse relations eventually leading to a wrong parse tree.

Also, change in domain leads to other problems including lexical sparsity and feature incompatibility among others. Strictly supervised models are unable to generalize to unseen lexical items, and this adds to the overall uncertainty of the system. Previous research [John et al., 2006; Huang and Yates, 2009; McClosky et al., 2010] has shown that the performance of the model falls significantly when the learned models are tested on different domains.

It is against this backdrop that we propose our thesis. As the volume of unstructured data is rapidly growing, various methods have been suggested in the literature that makes use of the abundant data by utilizing unsupervised machine learning techniques. One of the most significant contributions has been in the field of learning of unsupervised lexical representations (or embeddings)[1] [Turney and Pantel, 2010; Mnih et al., 2009; Collobert et al., 2011a; Turian et al., 2010a; Mikolov et al., 2013d; Pennington et al., 2014a]. The embeddings are commonly learned by exploiting the distributional property of words in a large corpus. The distributional hypothesis states that *words appearing in similar contexts have similar meanings and properties*.

In this thesis, we study the problem of using lexical embeddings in various NLP tasks by tailoring and manipulating embeddings for a given task. This is related to the application of semi-supervised learning to the task of NLP where unlabelled data is used to obtain information (such as word representations, etc..) and then use labelled data to learn a model for the task [Koo et al., 2008; Miller et al., 2004b]. This also follows parallel research in the field of NLP that uses lexical embeddings and has produced state-of-the-art results in many of the NLP tasks [Collobert et al., 2011a; Weiss et al., 2015; Chen and Manning, 2014a].

1.2 THESIS CONTRIBUTIONS

Our primary goal in this thesis is to explore and study techniques that make use of pre-trained word representations in different Natural Language Processing tasks. Specifically, our contributions are:

1.2.1 *Tailoring Word Embeddings for Bilexical Operators*

We address the task of learning functions that compute compatibility scores between pairs of lexical items under some linguistic relation. We call these functions as bilexical operators.

MOTIVATION. Word representations are usually generic lexical representations obtained on a large dataset. We study the problem of tai-

1 In this thesis we alternatively refer to embeddings as representations and vice-versa.

loring and compressing word representations for a task, given some supervised data.

SPECIFIC CONTRIBUTIONS:

- We propose a learning algorithm that takes an existing lexical vector space and compresses it such that the resulting word embeddings (or re-embedded representations) are good predictors for a target bilexical relation. Our proposed algorithm is formulated as learning a low-rank bilinear form and inducing low-dimensional embeddings of the lexical space tuned for the specific task.
- The low-rank constraint on the bilinear form results in computational advantages as the prediction now is expressed as the inner-product between low-dimensional embeddings.
- In experiments, we show that task-specific embeddings can benefit both the quality and efficiency of several lexical prediction tasks.

1.2.2 *Prepositional Phrase Attachment over Word Embedding Products*

We investigate the problem of resolving prepositional phrase attachment ambiguity using a binary and ternary prediction model.

MOTIVATION. The prepositional phrase attachment problem is a classic linguistic ambiguity problem that is still one of the main sources of errors for syntactic parsers. Classical approaches to resolving the ambiguity have used lexical, syntactic and semantic features amongst other features. Inspired by the implicit syntacto-semantic properties of word embeddings, we investigate the resolution of the ambiguity using word embeddings.

SPECIFIC CONTRIBUTIONS:

- We present a low-rank multi-linear model for the task of solving prepositional phrase attachment ambiguity.
- Our model exploits tensor products of word embeddings, capturing all possible conjunctions of the embeddings.
- Our results show that tensor products, compared to commonly used compositional methods (i.e., summing and concatenating), are better in performance and that a relatively simple multi-linear model that uses word embeddings of lexical features can outperform more complex non-linear architectures that exploit the same information.

- Our method also obtains significant improvements on out-of-domain tests by simply using word-embeddings induced from source and target domains.

1.2.3 *Vocabulary Expansion for Machine Translation by Mapping Embeddings*

We study the problem of resolving out-of-vocabulary words in the context of machine translation using a bilinear model that projects embeddings in two languages into a common subspace.

MOTIVATION. The presence of unseen words in either source or target side in Machine Translation systems can significantly affect the performance of the system. This might severely magnify erroneous translations. Several ways have been proposed that use lexical resources (dictionaries, morphological analyses, etc.) in the context of resolving poverty of lexical content.

SPECIFIC CONTRIBUTIONS:

- We introduce a method to obtain a probabilistic distribution of words in the target language for a given source word using a bilinear model that takes embeddings of words in both languages.
- Our model is relatively low-resource and generic and exploits only small source-to-target word dictionaries as supervision to map the embeddings in two different languages into a common subspace. This makes it extendable to other domains relatively with ease.
- In our experiments, we obtain consistent improvements in the translation quality especially on out-of-domain settings.

1.2.4 *Mapping Representations*

We consider the setting in which we train a supervised model neural network that learns task-specific word representations by back-propagating the errors.

MOTIVATION. We assume that we have access to some initial word representations (e.g., unsupervised embeddings), and that the supervised learning procedure updates them to task-specific representations for words contained in the training data. But what about words not contained in the supervised training data? When such unseen words are encountered at test time, they are typically represented by either their initial vectors or a single unknown vector, which often leads to errors.

SPECIFIC CONTRIBUTIONS:

- We address this issue by learning to map unseen words from initial representations to task-specific ones.
- We present a general technique that uses a neural network mapper with a weighted multiple-loss criterion.
- We consider the task of dependency parsing and report improvements in performance (and reductions in out-of-vocabulary rates) across multiple domains such as news, web-based data, and speech corpora.

1.3 THESIS OUTLINE

The structure of the thesis is as follows: Chapter 2 describes the background and previous approaches that use word embeddings. In Chapter 3, we describe our Low-Rank bilinear model. In Chapter 4, we discuss our work on resolving prepositional phrase attachment problem. We then describe our work on expanding vocabulary in the context of Machine Translation systems in Chapter 5. In Chapter 6, we describe our work on mapping word representations to task-specific representations. Finally, in Chapter 7, we conclude our work and describe current and future directions of research.

1.4 PUBLICATIONS

This dissertation summarizes several contributions to understanding and using word-embeddings in the field of NLP. Publications that are a direct result of this work include:

- Madhyastha et al. [2014] Learning Task-specific Bilexical Embeddings. P. Madhyastha, X. Carreras, A. Quattoni; In proceedings of International Conference on Computational Linguistics (COLING) 2014.
- Madhyastha et al. [2015] Tailoring Word Embeddings for Bilexical Predictions: An Experimental Comparison. P. Madhyastha, X. Carreras, A. Quattoni; In Proceedings of International Conference on Learning Representations (ICLR) 2015, workshop Track
- Madhyastha et al. [2016] Mapping Unseen Words to Task-Trained Embedding Spaces. P. Madhyastha, M Bansal, K Gimpel and K Livescu; In of the Workshop of Learning Representations for Natural Language Processing (RepL4NLP) 2016;

  In submission articles include:

- Madhyastha and España-Bonet [2016] Vocabulary Expansion for Machine Translation by Mapping Embeddings. P. Madhyastha, C. España-Bonet; Under Reivew;

- Prepositional Phrase Attachment over Word Embedding Products. P. Madhyastha, X. Carreras, A. Quattoni; Under Review;

During the course of this thesis, other collaborations have resulted in the following publications. These, however, are not detailed in this thesis summary:

- Quattoni et al. [2016] Structured Prediction with Output Embeddings for Semantic Image Annotation. A. Quattoni, A. Ramisa, P. Madhyastha, E. Simo-Serra, F. Moreno-Noguer. In Proceedings of the 15th Annual Conference of the North American Chapter of Association for Computational Linguistics: Human Language Technologies (NAACL:HLT) 2016

- Ellebracht et al. [2015] Semantic Tuples for Evaluation of Image to Sentence Generation. L. D. Ellebracht, A. Ramisa, P. Madhyastha, J. Cordero-Rama, F. Moreno-Noguer and A. Quattoni. In Vision and Language Workshop; In Proceedings of Empirical Methods on Natural Language Processing (EMNLP) 2015

- Costa-jussà et al. [2016] The TALP-UPC Spanish-English WMT Biomedical Task: Bilingual Embeddings and Char-based Neural Language Model Rescoring in a Phrase-based System. M. Costa-jussà, C. España-Bonet, P. Madhyastha, C. Escolano, J. Fonollosa. In First Conference on Machine Translation (WMT) 2016

2

BACKGROUND AND RELATED WORK

In this chapter, we briefly describe previous work on lexical embeddings. The chapter begins with a small introduction on representation learning in the field of NLP. We then survey word embeddings and a few of the most common unsupervised learning methods for obtaining word embeddings. We then discuss some methods, proposed in the literature, that compare word embeddings and some of the popular applications of word representations. We finally wrap the chapter with a discussion on challenges of using word embeddings and some of the recently proposed techniques and approaches that resolve these challenges.

2.1 A BRIEF OVERVIEW OF REPRESENTATION LEARNING IN THE FIELD OF NATURAL LANGUAGE PROCESSING

The performance of machine learning algorithms is directly proportional to the quality of features or data representation. An ideal representation should be capable of distilling all necessary and relevant information about the data such that, the machine learning algorithm is able to achieve state-of-the-art performance on a given task. Obtaining good representations is a challenging task, especially in the field of NLP. This is also sometimes referred to as the task of feature engineering. In general, a significant part of the effort is spent on feature engineering; usually, this involves data pre-processing, transformation of data and feature combination. Most of the state-of-the-art NLP systems depend on a combination of lexical features. Sometimes, obtaining expressive features for a particular task involve experience, prior knowledge and heuristics. Consider the task of part of speech tagging for English, some of the features used include knowledge about the capitalization of words, morphological information about words and lexical combinations of prior and posterior words among many more.

An increasingly growing literature of theoretical and empirical work suggests that traditional techniques for feature engineering in NLP are sometimes inefficient [Huang et al., 2013; Turian et al., 2010a]. This is especially true with the large set of domains in NLP, making features incompatible with a change in domain. One of the main reasons for this is because the lexical items follow a *Zipfian* distribution, this implies that there is always little training data for a substantial fraction of lexical items resulting in heavily sparse feature representations [Huang and Yates, 2010; Bikel, 2004]. This effectively hurts the

machine learning algorithm's ability to generalize to unseen lexical items and in turn affecting the performance of the system.

The area of representation learning, to an extent, tries to solve the problem by automatically learning necessary and relevant representation of the data. In turn, this makes it possible to extract relevant and useful information about the data when building predictors for different tasks. Bengio et al. [2013] give a thorough survey of recent representational learning frameworks. According to Bengio et al. [2013], an ideal representation should possess following properties among others:

(a) Expressiveness: A good representation should be able to distil all necessary task related properties. A carefully engineered feature representation is often specific and mostly contains task specific properties. While the unsupervised representations preserve many properties, however, they may not be the best performing for all kinds of tasks.

(b) Compactness: An ideal representation should be able to contain all task-relevant properties in a compact low-dimensional vector. In particular for NLP tasks, there is a possibility of having a large number of lexical units, having low dimensional representations will be useful.

(c) Abstraction: The representation should be able to capture abstract properties such that any change in the domain or data would resist changes in properties of the representations in a way that the representations become less efficient or less powerful. However, the abstraction is a two-edged sword. Heavy abstraction may prove to be unusable for many tasks, while heavy specification might make stubborn to changes in the domain.

An ideal representation, in general, reduces the computational burden of performing classification or prediction related steps. It has been observed that good representations seem to generalize better to features that co-vary the most with respect to the outcomes of the task.

More specifically, the goal of representation learning is to learn some underlying structure of the data. However, one of the biggest challenges of representation learning is that it's hard to establish a clear objective or target for training such that the representations are useful over a general set of tasks.

2.2 LEARNING WORD EMBEDDINGS

Vector representation of words (also referred alternatively as word vectors, word embeddings, word representations in the literature) is based on capturing some relationships between words broadly based

on co-occurrence properties of the words in a language. They espouse the unifying philosophy that the words that occur in similar contexts tend to have similar properties. For example, the words 'good' and 'nice' have similar meanings as they are, in many cases, replaceable by one another when used in the same context. Word vectors try to capture the distributional properties by using the co-occurrence patterns. In general, all of the word vector based models can be approximated to be built on some notion of semantic similarity. There are many different ways of producing a computational model of semantic similarity; however, the underlying theory and the assumptions are similar. The neologisms in this field might create confusion even if the underlying ideas are mostly the same. In the following sections, we will try to review basic concepts related to word embeddings.

Formulation

Suppose that we are given a sufficiently large corpus of unannotated text in a particular language. Let $\mathcal{V}$ be a vocabulary, and let $w \in \mathcal{V}$ denote a word. We want to find a function ϕ, such that:

$$\phi : \mathcal{V} \rightarrow \mathbb{R}^n \tag{2.1}$$

where $\phi(w)$ is the n-dimensional representation of word w. In this chapter we use $\phi(w)[i]$ to refer to the i-th coordinate of the vector.

In the field of NLP, the data is structured, consisting of trees, sequences, phrases, etc.. Ideally, a feature representation should be able to distill regularities in such structured data and model some form of similarities between the latent structures. Lexical representations have some of these properties, which make them suitable for use in statistical models.

Typically, in previous literature [Sahlgren, 2006; Turney and Pantel, 2010] similarity has been discussed in the context of

- Substitutional or Paradigmatic similarity -two words are similar if in they are replaceable in some context. Eg., *make* and *create*.
- Syntagmatic Similarity — two words are similar if they occur together in some text. Eg. *eat* and *pizza*

2.2.1 *Geometric Interpretation*

To briefly understand the geometric interpretation of lexical representations we here take a simplistic example of vectors in a two-dimensional euclidean space. We illustrate the geometric interpretation of Euclidean distance correlating with the concept of similarity using Figure 2.1. In Figure 2.1 we are given words *sun*, *moon* and *dog* with contexts as *shadow* and *shine*. We notice that the Euclidean

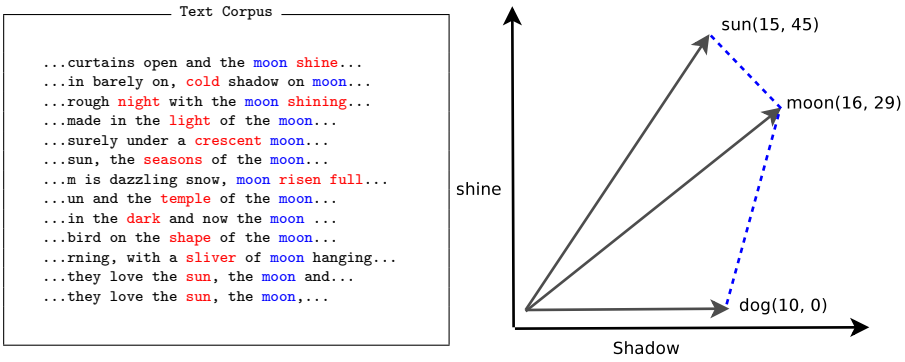


Figure 2.1: Geometric Interpretation of Word Vectors (based on Baroni [2012]): Here axes *shine* and *shadow* are the context words and the three target words are *sun, moon* and *dog*. The vectors are formed using by using context words in the corpus as shown in the left.

distance of *sun* and *moon* is much smaller than the Euclidean distance between either of the words and *dog*. This closeness in the Euclidean distance, in the given space, can help us in inferring words with similar properties. In the example, *moon* is similar to *sun* under the specified distribution (here the context as *shine* and *shadow* — a two-dimensional space).

In the following subsections, we will refer to similarity mostly in the context of substitutional aspects. We will now survey some methods of producing word representations. There are many ways of factoring the representations based on the way they are learned. In this thesis, we base this loosely on the structure followed by Baroni et al. [2014].

We first describe count-based models. Some of the ideas from count-based models are the recurring themes in several modeling techniques. We discuss some of the popular methods that implement the principles of count-based models. We then briefly review previous work on clustering based approaches and some of the popular prediction based modeling techniques.

2.2.2 *Count-Based Models*

The principle idea behind count-based models is utilizing the statistics of word occurrences in a certain context. They are loosely based on the distributional hypothesis [Turney and Pantel, 2010], which states that statistics of word usage can be used to obtain measures of meaning. The models built under the hypothesis typically make use of the word co-occurrence in the form of matrices hence, exploiting the latent structure of the co-occurrence matrix. Let us begin with some notations. Let the matrix of word co-occurrence counts be denoted by X, the entries X_{ij} of which mean the number of times word

j is seen in the context i (or vice versa, the choice should ideally not change any representational aspects). We will expand on the definition of context below. X is mostly a sparse matrix. Typically, there are many ways of associating contexts in count-based models, we survey some of the popular methods:

- Word-Document Matrix: In a Word-Document matrix, typically we have the contexts as the documents. So we effectively use the count of occurrence of a word in a document. Words are rows and the documents are columns of these matrices. This gave rise to *Bag-of-Words Hypothesis*, that is, the frequency of a set of query words in a document indicates the relevance of the document to a query [Salton and Buckley, 1988]. Here, $X_{:j}$ corresponds to the bag-of-words for a document. Further, Deerwester et al. [1990] observed that by looking at the word vectors, here row-vectors i.e., $X_{i:}$, we can measure word-similarity.

- Word-Context Matrix: This method was originally inspired by the *distributional hypothesis*, from classical linguistics, which states that words that occur in similar contexts have similar similarity measures [Harris; Firth, 1957; Deerwester et al., 1990]. A word context matrix is similar to the word document matrix in many respects. The most distinguishing factor is mostly the context, here the context is some form of a lexical item. Some of the popular contexts used in literature include:

  Windows of words: For a target word, a window of k-words before and after the words is used [Lund and Burgess, 1996]. This is also sometimes referred to as Continuous Bag-of-Words model as the left and right contexts can be considered as a set of words in a bag of window k. One of the drawbacks of such a method is that it loses any positional information about the context words with respect to the target word. The two most popular techniques for extracting the context commonly uses n-gram based technique where the context is considered over the bag-of-words in a given window. The other common technique is skip-gram [Guthrie et al., 2006] where words are considered with l-skips over n-grams, where $l < n$.

  Grammatical Contexts: Here, contexts are defined as some notion of grammatical property given a target word [Lin, 1998; Lin and Pantel, 2001; Pado and Lapata, 2007; Grefenstette, 1994]. Usually, these methods make use of syntactically parsed data to build contexts that reflect properties such as dependency relations between the words. These are sometimes referred to as structural contexts as they involve the use of some structural information. Hindle [1990] used predicate-arguments as the structural context. Schütze

and Pedersen [1995] used directional information i.e., explicit directional information with words as contexts. Sahlgren [2006]; Wiemer-Hastings and Zipitria [2001] use parts of speech information along with words as contexts.

- Pair Pattern Matrix: Lin [1998] describe pair-pattern matrix as a matrix where the rows refer to pairs of words that are related by some relation. For example, *mason:stone* and the columns are patterns where the pair occurs, for example, *mason makes use of stone to ...* — in this case, the pattern 'makes use of' is the pattern for the pair 'mason' and 'stone'.

  One of the well-known problems of using count-based representations is the issue of the ideal word context. There is no single best way of defining the word context matrix [Pado and Lapata, 2005]. In many cases, some preprocessing is performed on the corpus that includes lemmatization, stemming, etc. Some preprocessing techniques, in particular for methods that involve linguistic information are non-trivial and sometimes may need extra-linguistic resources.

Now, we shall focus on some of the very popular methods in which these word-contexts are used to get suitable word embeddings.

2.2.2.1 *Hyperspace Analogue to Language (HAL)*

HAL [Lund and Burgess, 1996] uses word co-occurrence matrix with a window of size k such that there are k-words to the left and k-words to the right. The co-occurrences are weighted by the distance between the target word and the contexts. The resultant matrix is a semi-directional co-occurrence matrix. This produces a very high-dimensional context vector. The HAL typically reduces the dimensionality by discarding columns with the lowest variance, thereby, considering only the top-d most variant vectors. This, however, may not be essential, but it has been observed that the most variant columns tend to dominate in the computation of the Euclidean distance between two vectors. The Euclidean distance ranks vectors with increasing measure but decreasing semantic similarity (i.e., the more similar the two vectors are, smaller the distance between the two vectors). One of the main problems with HAL is that the contribution from most frequent words is higher than other words: for example, co-occurrence with stop words such as 'the', 'is', etc., will have a large effect on their similarity in spite of contributing very little to semantic relatedness. COALS method is one of the techniques [Rohde et al., 2006], where the co-occurrence matrix is transformed by a correlation-based normalization. This results in an even distribution of the counts.

2.2.2.2 *Latent Semantic Analysis (LSA)*

LSA [Deerwester et al., 1990; Landauer et al., 1998] was originally based on the word-document matrix, but the principle can equally be applied to word-context matrices and pair-pattern matrices. The basic approach is building a word co-occurrence matrix X followed by normalization of the counts. This is followed by computation of singular value decomposition (SVD) of the normalized co-occurrence matrix. The SVD is a product of three matrices, U containing orthonormal columns known as the left singular vectors, S — a diagonal matrix containing the singular values and $V^\top$ containing orthonormal rows referred to as the right singular vectors. The left and the right singular vectors are also usually seen as eigenvectors and the singular values as eigenvalues. The singular vectors reflect principal components or axes of greatest variance in the data. Typically the matrices comprising the SVD are permuted such that the singular values in S are in decreasing order, they then can be truncated to a much lower rank,r. The product of these truncated-matrices is the rank-r approximation of the X. The similarity between two words in LSA is usually computed using the cosine of their reduced dimensionality vectors. A truncated SVD applied to word-document matrix is referred to as Latent Semantic Indexing (LSI). It has been observed in Deerwester et al. [1990] that dimensionality reduction improves the quality of word embeddings for various tasks. Also, it has been observed that taking a square root [Rohde et al., 2006] or logarithm [Church and Hanks, 1990] before SVD can improve the quality of the representation.

It has been observed that the truncated SVD results in capturing latent interactions of the word representations. Although, in hindsight, dimensionality reduction is expected to induce some noise and hence the quality of the word vectors is supposed to reduce; however, it has been observed that in most of the cases the quality of the vectors have improved. In a recent work by Arora et al. [2016] they prove that the noise for a semantic vector space is reduced by about $\sqrt{(r/n)}$ by doing dimensionality reduction (here r is the reduced dimension and n is the original dimension). Also, Arora et al. [2016] argue that dimensionality reduction improves the quality of word embeddings for various tasks is because of the directionless property of word vectors. This has a *purification* effect.

Bullinaria and Levy [2007, 2012] discuss different set of factors to extract proper semantic representations from word co-occurrence statistics from large text corpora.

2.2.2.3 *Non Negative Sparse Embedding (NNSE)*

NNSE is a technique proposed by Murphy et al. [2012] where given an input word context matrix, the method produces a latent sparse

representation using matrix factorization technique. NNSE solves this objective function:

$$\arg\min \sum_{i=1}^{w} \|X_{i:} - A_{i:} \times D\|^2 + \lambda |A|_1$$

such that: $D_{i:} D_{i:}^{\top} \leqslant 1; Aij \geqslant 0$. This will find $A \in \mathbb{R}^{v \times d}$ that is sparse and non-negative and $D \in \mathbb{R}^{d \times n}$ contains the corpus statistics in latent space. Non-negative representations have often been shown to have better interpretability. Similar methods have been recently explored [Yogatama et al., 2015; Yogatama and Smith, 2014] that explicitly produce sparse representations given a dense word vector.

2.2.2.4 *Helinger PCA*

Lebret and Collobert [2014] introduce a way of learning embeddings that is related to Latent Semantic Analysis. They use similar kinds of word-context matrices as described in previous section. However, here instead of counts, they consider co-occurrence probabilities, i.e.,:

$$\Pr(w|c) = \frac{n(w,c)}{\sum_c n(c,w)}$$

where: w is the target word and c is the context word. In essence they obtain the word co-occurrence probability matrix. This is then followed by a square root operation resembling Hellinger distance for discrete probability distributions. Further, dimensionality is reduced by considering a principal component analysis over the resultant square rooted word co-occurrence probability matrix. They also mention that increasing the context window helps to capture better syntactic and semantic information about the words.

2.2.2.5 *Muti-View Learning Based Approaches*

Dhillon et al. [2015] introduce a method that is based on canonical correlation analysis (CCA) for learning word embeddings. First, they construct two matrices — one for left word co-occurrence matrix that considers left contexts and another right word co-occurrence matrix that considers right contexts. They initialize a matrix A that is $v \times k$ (where v is the vocabulary of the corpus and k is the required dimensionality). They project the left and right context matrices and recursively compute the CCA by using reduced rank left and right context matrices to compute the matrix A bounded by certain error. This essentially maps each word to a reduced rank k-dimensional state vector. Further, they use CCA between the hidden states and the token matrix to induce context specific embeddings for the tokens. Dhillon et al. [2015]'s method basically exploits the left and right similarity by alternatively estimating word state by using CCA and averaging over the states with all occurrences of the word. Another related approach

by Stratos et al. [2014] uses CCA by using one-hot representations of words and one-hot representations of contexts and projecting these word and context matrices to a low-dimensional space in which they would be maximally correlated. The projected word matrix is considered as a new word representation.

2.2.2.6 *Pointwise Mutual Information Based Models*

Pointwise mutual information is an information theoretic approach to finding collocations. Basically, it is a measure of how much information is conveyed by some word given another word. Pointwise Mutual Information (PMI) based models have been extensively used in the literature [Turney and Pantel, 2010; Lapesa and Evert, 2014; Turney, 2001; Baroni and Lenci, 2010]. However, PMI might be a problem when the count of $(w, c) = 0$ and hence most of the early work in literature circumvent this problem by considering a Positive PMI (or PPMI). For a given corpus with vocabulary v, PPMI for word w given a context word c is given as:

$$\begin{aligned} \text{PPMI}(w, c) &= \max\left(\log\left(\frac{\Pr(w, c)}{\Pr(w)\Pr(c)}\right), 0\right) \\ &= \max\left(\log\left(\frac{\sum(w, c) \times \|v\|}{\sum_{\acute{c}}(w, \acute{c}) \times \sum_{\acute{w}}(\acute{w}, c)}\right), 0\right) \end{aligned}$$

This is then used to make a co-occurrence matrix whose components are $\text{PMI}(w, c) \forall c$. Complicated models consider asymmetric PMI matrices with context words with term re-weighting.

Levy and Goldberg [2014a] define special cases of PMI — a. shifted PMI as $\text{SPMI}(w, c) = \text{PMI}(w, c) - \log(k)$ for some $k \in \mathbb{R}$ and b. shifted positive PMI as $\max(\text{SPMI}(w, c), 0)$. Usually, a method similar to LSA is applied on PMI based co-occurrence based matrices to reduce the dimensionality.

2.2.3 *Clustering Based Approaches*

Clustering-based approaches typically induce clustering over words. These approaches partition sets of words into clusters or subsets of similar words. The number of unique clusters is always lesser than the vocabulary in any dataset. In most cases, the clustering based approaches that use unlabeled data are based on the syntagmatic similarity.

2.2.3.1 *Brown Clustering*

The brown clustering algorithm [Brown et al., 1992] is a popular algorithm and has been widely used in NLP in Miller et al. [2004a]; Liang [2005]; Koo et al. [2008]. It is an agglomerative clustering algorithm

that generates hard clustering. Broadly the algorithm is based on co-occurrence of the words. Given a corpus of text, initially, each word is in its own cluster. In the following steps, the algorithm iteratively merges the pairs of clusters that minimize the likelihood of the text corpus based on class-based bigram language model defined on word clusters. This recursive operation gives rise to hierarchical clustering of words. The hierarchy of words is useful as word classes are chosen at different levels of hierarchy.

Ushioda and Kawasaki [1996] present an extension to the brown clustering algorithm by learning hierarchical clustering of phrases. Further, Martin et al. [1998] present extensions of the brown clustering algorithm that considers bigram and trigram statistics. Uszkoreit and Brants [2008] present predictive exchange algorithm, which is an extension to the brown clustering algorithm, in that, they consider class conditioned words, instead of classes conditioned on classes. The authors mention that word-to-class transitions statistics can directly be obtained while clustering large data sets.

2.2.3.2 *Distributional Clustering*

Pereira et al. [1993] use a word co-occurrence matrix initially and then transform the matrix using clustering approaches. They preprocess the data to collect two categories of words — verbs and nouns. They cluster nouns according to their conditional verb distributions. The basic idea is focused on minimizing the average similarity (as Kullback-Leibler divergence) between $\Pr(\mathit{verb}|\mathtt{noun})$ and noun centroid distributions. Baker and McCallum [1998] apply the distributional clustering scheme of Pereira et al. [1993] for clustering words represented as distributions over categories of the document entries where they appear. Given a set of c categories and a distribution of word given the categories $\Pr(w|\mathtt{c})$, the words are clustered by an agglomerative clustering algorithm.

2.2.4 *Prediction Based Models*

In the last few years, there has been a large exploration of prediction based models. Essentially, prediction based models formulate the task of learning word representation as a pseudo-supervised task, i.e., essentially the models maximize the probability of the word-context occurrence in the corpus. This, in principle, can be seen as the reverse of count-based models. In the case of prediction based models, the vectors are learned such that it optimally predicts the correct word-context association, whereas in the count-based models the counts are collected before and then reweighted. In essence, these models represent words as dense, low-dimensional vectors of real numbers. These models are also known widely as *distributed representations* in literature. The idea of distributed representations could be traced back

to Hinton [1986]. Typically, a training algorithm tries to push words that are functionally similar to be replaceable. This makes the training algorithm learn word features that relate to a form of semantic or syntactic similarity. The supervision in most of the cases is based on negative sampling or contrastive estimation [Smith and Eisner, 2005], i.e., exploiting implicit negative evidence. This technique is computationally efficient and is robust. Prediction based models that use neural network architecture are also referred to as neural word embeddings.

In the following subsections we survey some of the popular techniques in the literature:

2.2.4.1 *Language Model based Embeddings*

Representing words as dense vectors, first introduced by Bengio et al. [2003], had the goal of improving standard n-gram language models on statistical language modeling tasks by using distributed representations. The log-bilinear model by Mnih et al. [2009] follows the idea and proposes a model such that, when given a corpus, it takes n-gram sequence and linearly combines the representations of the $n-1$ words. It then learns a log-bilinear model to predict the embedding of the correct last word. That is, let each word be represented by a d-dimensional vector v, then:

$$v_{w_{c(w)}} = \sum_{i=1}^{n-1} H_i v_{w_i}$$

where, H_i is a parameter matrix (the weights) and v_{w_i} are real valued d-dimensional vectors.

The distribution for the next word then is computed based on the similarity between the predicted representation and the representations of all words in the vocabulary:

$$\Pr(v_{w_n} | v_{w_{c(w)}}) = \frac{\exp(v_{w_{c(w)}}^\top v_{w_n})}{\sum_{j \in v} \exp(v_{w_{c(w)}}^\top v_{w_j})}$$

Mnih et al. [2009] speed up model evaluation during training and testing by using a hierarchy to exponentially filter down the number of computations that are performed. This hierarchical technique was first proposed by Morin and Bengio [2005]. The technique exploits 'hierarchical' binary trees where words are associated with leaf nodes with each leaf node as one word. If the tree is balanced then each n-way decision can be replaced by $O(n \log n)$ binary decisions for predicting next word, thereby achieving an exponential speed-up. This model, combined with the optimization technique, is called the hierarchical log-bilinear HLBL model.

2.2.4.2 *SENNA Embeddings*

Collobert et al. [2011a] presented a neural network based approach to learn lexical representation called senna, which is discriminative and non-probabilistic. At each training step, it reads n-grams of words from a given corpus. We can represent each word as a d-dimensional vector representation v_w. As in HLBL, they concatenate the all word vectors for the n-gram representation:

$$v_{w_{c(w)}} = \sum_{i=1}^{n} H_i v_{w_i}$$

They then propose to have negative examples by corrupting one of the word in the n-gram. The corrupted word is chosen uniformly from the vocabulary and the vectors are linearly added in a similar way to get $v_{\tilde{w}_{c(w)}}$ It is essentially a single layer neural network — so both $v_{w_{c(w)}}$ and $v_{\tilde{w}_{c(w)}}$ are passed through the neural network to obtain $s(v_w)$ and $s(\tilde{v}_w)$. The model uses margin-based loss such that:

$$\arg\max(0, 1 - s(v_w) + s(\tilde{v}_w))$$

That is, to have the score of the correct n-gram score to be higher than the corrupted n-gram score. The model uses stochastic gradient descent to minimize the loss over n-grams in the corpus. Bengio [2009] use a similar approach, with the difference of the corrupted word. The corrupted word is different — in Collobert et al. [2011a] the middle word in the n-gram is corrupted, while Bengio [2009] corrupt the last word.

2.2.4.3 *Skip-gram based word vectors using Negative Sampling*

Skip-gram embeddings, introduced by Mikolov et al. [2013d], are computationally efficient models for obtaining distributed representations from a large amount of text. Given a corpus of vocabulary v, for a target word w and a context word $c \in C$, all words are represented with a d-dimensional vector representation $v \in \mathbb{R}^d$. They describe the probability of context given the target word as:

$$\Pr(v_C|v_w, \theta) = \prod_{c \in C(w)} \Pr(v_c|v_w, \theta)$$

That is, the model is trying to seek parameters θ such that the dot product $v_c^\top v_w$ for every 'correct' word-context pair is maximized. They do it in two ways, by contrastive estimation or negative sampling, the model collects samples where w and c co-occur as word contexts in the data (also known as 'correct' samples) and generates a sample where w and c do not co-occur as contexts by randomly sampling all the contexts or by using a standard maximum entropy loss. The distribution is modeled as:

$$\Pr(v_c|v_w, \theta) = \frac{\exp(v_w^\top v_c)}{\sum_{c' \in C(w)} \exp(v_c^\top v_w')}$$

The objective is to maximize the log-probability of the observed correct pairs:

$$\arg\max_{\theta} \prod_{v_w \in \text{Corpus}} \prod_{c \in C(w)} \Pr(v_c | v_w, \theta)$$

The objective is trained using stochastic gradient descent over all v in the corpus. Maximizing the objective results in observed word-context pairs to have similar word vectors. The words could be assumed to be replaceable in similar contexts [Mikolov et al., 2013d]. The contexts in skip-gram based model are chosen based on l-skips in the n-gram bag-of-context-words.

Mikolov et al. [2013d] also introduce a related model called Continuous Bag of Words model that uses bag-of-words based contexts to model $\Pr(v_w | v_C)$. However, in this case, the model maximizes the probability of occurrence of the word for some 'correct' context word c. The model uses a similar machinery as before, but it uses bag-of-words for extracting contexts.

The authors mention that the skip-gram based model being effective for a relatively smaller corpus and the continuous bag-of-words model obtain better results for a sufficiently large corpus.

2.2.4.4 *GloVe Embeddings*

GloVe [Pennington et al., 2014a] is another unsupervised learning algorithm which is trained, based on a PMI based matrix. This algorithm, unlike others that have been described here, starts by first building a matrix that resembles positive pointwise mutual information and uses a log-bilinear model of word and context by minimizing a least square objective constrained by the PMI based matrix. Arora et al. [2016] shows that, under some circumstances, GloVe produces similar vectors as skip-gram based vectors as seen in the previous subsection.

2.2.5 *Association between Prediction and Count-Based Methods*

It has been shown that some of the neural embeddings based methods are closely related to the count-based PMI models in recent exploration by Levy and Goldberg [2014a] and Arora et al. [2016]. Levy and Goldberg [2014a] show that by accumulating data over co-occurrences of words w and contexts c, the objective function of skip-gram with negative sampling can be written as shifted point-wise mutual information and show that under some conditions, the skip-gram with negative sampling objective is factorizing a count-based shifted PMI word-context matrix. Further, Arora et al. [2016] describe a probabilistic model of text generation that augments the log-linear predictive models with random-walk over a latent discourse space. They show

the relation between the count-based models and negative sampling based skip-gram models.

2.3 COMPARISONS AND APPLICATION OF WORD REPRESENTATIONS

We have discussed various ways of creating word vector representations. In this section, we begin by discussing challenges of comparing word vectors. One of the substantial problems with word vectors is that there is no standard method for evaluating the quality of the word vectors. This is attributed to the difficulty in interpreting the word vectors, that is, there is no standard way to interpret each dimension in a word vector and compare with different representations. Also, it is not completely clear how the representations can be used in various NLP tasks for best returns. We then discuss some of the applications of these word representations in some of the state-of-the-art methods.

2.3.1 *Comparison of Lexical Representations*

In this subsection, we list the most frequently used techniques to compare various word representations.

2.3.1.1 *Lexical Similarity Based Metrics*

Word-similarity metrics are used to measure how well the representations capture word-similarity in the form of 'replaceability'. These datasets include WordSim353 [Finkelstein et al., 2001], rare words dataset [Luong et al., 2013a], WordSim203 [Agirre et al., 2009], WordRel252 [Agirre et al., 2009], semantic similarity datasets [Miller and Charles, 1991], noun dataset [Rubenstein and Goodenough, 1965], frequent words dataset [Bruni et al., 2014], MTurk287 [Radinsky et al., 2011], Mturk-771 [Halawi et al., 2012], Verbs dataset [Yang and Powers, 2006] and Sim-lex99 [Hill et al., 2015]. The standard procedure to compute similarity is to compute cosine distances between word pairs and rank these. This is followed by computing the Spearman's rank correlation [Hauke and Kossowski, 2011] between the model calculated ranking and the human ranking. However, most of these datasets have problems related to low inter-annotator agreement and sometimes many dissimilar words receive high agreement, etc.. These cause some of the word representations to surpass the inter-annotator agreement ceiling. A more comprehensive analysis of the datasets can be found in Hill et al. [2015]. It has been observed that in general, the larger the initial corpora used for obtaining word representations, the better is the performance of the representations on these datasets. We also observe that the LSA-based and PMI based mod-

els and the Neural Embeddings are equally competitive in some of these datasets. Baroni et al. [2014] perform an extensive evaluation on some of the previously mentioned datasets. They perform experiments with count-based models and prediction based models and find that more recent prediction based models in most cases, outperformed the count-based models. However, Hill et al. [2015] also perform extended experiments using concept based datasets, part of speech based fine-grained similarity datasets and similarity versus association based datasets and conclude that count and prediction based models are equally competitive.

There have been other work that evaluates the count-based word vectors with other approaches, this includes, the comparison between count-based word vectors and `wordnet`-based approaches [Agirre et al., 2009]. Some work also evaluates the effect of window sizes on word vector representations on these tasks [Levy and Goldberg, 2014a; Hill et al., 2015; Chen et al., 2015; Bansal et al., 2014a; Bansal, 2015].

2.3.1.2 *Lexical Analogies*

This task was specifically introduced in Mikolov et al. [2013d] to evaluate syntactic and semantic relations between words. In each task, one of the words is missing and the task is to predict the correct missing word given the relation between another pair of words. In Baroni et al. [2014] it is observed that the prediction based models are better at performing these tasks.

2.3.1.3 *Performance of Embeddings in Tasks*

One of the ways of comparing the representations is by evaluating their utility in downstream applications. Towards this effort, there have been some recent works that use different types of word embeddings for evaluation. Turian et al. [2010a] performed extensive experiments with word embeddings on word chunking and named entity recognition tasks. They observe that concatenating different types of representations help improve the performance of the algorithms on both of these tasks. This is because count-based representations, clustering based representations, and prediction based representations inherently produce different kinds of errors.

Andreas and Klein [2014] study the performance of embeddings on the task of constituency parsing. They explicitly add word embedding based information to the features of the constituency parser especially in the context of out-of-vocabulary words. They note that given adequate training data to the parser, the embeddings do not add a lot of information, and the improvements are minimal to modest.

Schnabel et al. [2015] perform experiments with different kinds of word embeddings on a set of tasks including word relatedness, co-

herence, and downstream performance. They show that embeddings behave differently on various tasks.

In general, we notice an inconclusive trend with respect to the evaluation of the word embeddings.

2.3.2 *Application of Word Representations*

In the recent years, natural language processing and related fields have made use of word representations in a variety of tasks. Word vectors have been ubiquitously used for word-similarity based tasks ever since Deerwester et al. [1990]'s work, which showed that similarity could be measured by using word vectors. A great amount of literature follows this work by using different types of word vectors on lexical similarity based tasks [Landauer et al., 1998; Lund and Burgess, 1996; Schütze and Pedersen, 1995; Lin and Pantel, 2001] that includes discovering synonyms, antonyms, plurality, etc.

In areas of research involving information extraction or question answering, etc., identifying the existence of relations is an important challenge. Previous work [Turney, 2006; Jurgens et al., 2012] use word vectors to measure relational similarity and it has been seen as a flexible and straightforward solution that is competitive and most often performs better than using an exclusive relational classifier for a predefined set of relations. Turney [2006] use word vectors based on pair-pattern matrices on multiple-choice analogy questions from SAT college entrance test and achieve human-level performance.

Word Sense Disambiguation (WSD) is another important task where word vectors are successfully utilized. The oldest Application of word vectors to the task was shown in Leacock et al. [1993] where they use word-context frequency matrix directly for the task of WSD. Here, instead of using a word directly, a word with a sense tag was used. Further word vectors have been used to enhance a state-of-the-art WSD methods [Yuret and Yatbaz, 2010]. New methods tailor word vectors to contain sense specific information [Cheng and Kartsaklis, 2015; Tian et al., 2014; Huang et al., 2012a]

Word embeddings have been used for the task of named entity recognition both as additional features as well as main features [Lin and Wu, 2009; Turian et al., 2010a]. Collobert et al. [2011a] use word embeddings in a multi-task learning framework. Collobert and Weston [2008] propose a joint task of chunking, named entity recognition and part of speech tagging jointly and use similar representational information.

Syntactic Parsing is another key area using different types of word representations. Koo et al. [2008] use brown clusters along with the standard linear features to get a boost in the parsing accuracy. They also conclude that clustering information gives a significant performance improvement when we have limited supervision. Cirik and

Sensoy [2013] explore the usage of word representations for multilingual parsing. Bansal et al. [2014a]; Bansal [2015] convert distributed representations to link embedding or binary clustering information and feed it to the dependency parser and achieve very competitive results. Further, Chen and Manning [2014a]; Lei et al. [2014]; Weiss et al. [2015] use direct word embeddings as lexical features and achieve state-of-the-art results in parsing.

Maas et al. [2011]; Socher et al. [2013c]; Tai et al. [2015] use word vectors for sentiment analysis tasks and achieve state-of-the-art results. Sutskever et al. [2014]; Bahdanau et al. [2015]; Gao et al. [2013]; Devlin et al. [2014] use word embeddings for the task of machine translation and achieve competitive scores. Köhn et al. [2014] use word embeddings in the context of morphological prediction.

Word representations have also been used extensively in related fields that combine NLP and image processing. One of the most prominent approaches is by using word representations for zero-shot learning. Zero-shot learning consists of learning to recognize new concepts by just having a description of them. Frome et al. [2013] introduced a method that uses word representations to get a visual-semantic model. Socher et al. [2013a] employ a linguistic model as an intermediate semantic representation using word representations.

2.4 CHALLENGES AND EXTENSIONS

In this section, we will explore some of the major limitations and some proposed techniques to overcome these limitations.

2.4.1 *Polysemy, Lexical Substitution, Lexical Entailment*

Some words have more than one sense. These words are known as polysemous words. The most word embeddings assume a single vector per word type thus ignoring polysemy, which might be adding to the noise in the downstream task. That is, say a word such as 'work' as a noun and as a verb have same word vectors. In high dimensional spaces it could be argued that a vector is close to multiple regions, this still results in words that are synonyms with different senses to be clustered close together.

Another major limitation of the unsupervised methods is that the words are represented as a single point in space. This results in rigid vectors that do not exhibit properties of entailment. For example, 'Music' and 'Rhythm', in some cases, 'Music' entails the concept of 'Rhythm'. This entailment is difficult to capture in the traditional word embedding methods.

2.4.1.1 *Extensions to the Existing Vector Representations*

Previous work tries to engineer senses into embedding methods. Previous work by Reisinger and Mooney [2010] pre-cluster contexts of word tokens into discriminated senses and then use these clusters to re-label the tokens according to word sense. They then use the sense tagged corpora to learn embeddings. Huang et al. [2012a] improve the method by using an earlier version of single type-level word embeddings as contexts. Both these methods include fixed number of sense per word type. There have been extensions to these methods [Weston et al., 2013; Neelakantan et al., 2014], however, their coverage of senses is limited to only those senses that are covered in the underlying corpus.

Most previous related research uses multiple points in space to represent the different senses of words. But, the relations between words are not explicitly captured. Vilnis and McCallum [2015] describe a novel model that uses Gaussian distribution as a representation and obtain density based distributed word embeddings. They report many interesting advantages, which include better-capturing uncertainty about a word and its relationship with other words and modeling asymmetric similarities among others.

2.4.2 *Task Specific Embeddings*

Word representations learned using both parametric and non-parametric methods, while powerful, are not learned exclusively for a single task. These embeddings are highly abstract representations and mostly are learned to maximize the similarity (Euclidean or otherwise) under a certain context. In general, to be effective in NLP tasks, they should capture semantics relevant to the NLP task.

2.4.2.1 *Extensions to Existing Vector Representations*

Most word embedding methods are fast to train and can be readily downloaded. There has been a variety of recent work in the field of representation learning that try to task-specify word embeddings. Labutov and Lipson [2013] propose a technique that takes input representations with some labeled data and outputs an embedding in the same space but with properties that cater better for the task at hand. Astudillo et al. [2015]; Rothe and Schütze [2015] extend the technique for various tasks. Lebret and Collobert [2014] initialize neural networks with word representations and specify for a task with some labeled data by back propagating the errors for the task loss, thereby task specifying the representations. They note that the task specified representations perform better on a variety of tasks.

Some methods learn embeddings for a task jointly with the unlabeled data as a multi-task learning approach. Collobert et al. [2011a]

have successfully applied this approach that learns task-specific embeddings in this fashion. However, this is hard and takes a long time for the embeddings to be learned and specified and is computationally expensive. Also, it has a downside of overfitting or underspecifying words that are rarely seen in the supervised corpus, in many cases substituting a default word embedding for any rare word. This usually results in noise. Related methods such as Tang et al. [2014]; Maas et al. [2011] use a similar approach for the task of sentiment analysis.

Recent work also tries to replace the linear context within a sentence with a syntactic one [Bansal et al., 2014a; Levy and Goldberg, 2014b] for the task of dependency parsing. They use skip-gram with negative sampling based technique to learn the word embeddings. Here, a huge corpus is first parsed using a state-of-the-art dependency parser and the context of a word is taken to be the words that are in its proximity in the parse tree, together with the syntactic relation by which they are connected. These are then used as additional features for the task of dependency parsing, but here the word embeddings are further clustered, and cluster labels are used as features.

Another related line of work [Gouws and Søgaard, 2015] uses simple dictionary based technique to learn task-specific bilingual word embeddings that can be useful for cross-language knowledge transfer. Faruqui et al. [2015a] propose a method that refines word representations using relational information obtained from lexicons. They try to optimize the linked words in the lexicon to have similar representations.

2.4.3 *Representational Space and Models*

Another key challenge with using word representations in NLP models is the compatibility of word representation space with the existing state-of-the-art machinery. Most of the representations cannot be directly used as features for supervised linear models. While, this is not a big concern for neural network based models, as they update using error back propagation, they may not be portable easily to the linear models.

2.4.3.1 *Extensions*

Brown-clusters have been effectively used for different tasks including named entity resolution [Miller et al., 2004a; Liang, 2005] and syntactic parsing [Koo et al., 2008] and have produced state-of-the-art results. Qu et al. [2015a] perform a battery of NLP tasks with different word representations that include neural word embeddings, brown clustering, etc., they observe that brown-clusters perform as good as state-of-the-art or better. Brown clusters, by virtue of the sparse low dimensional binary vectors can be easily included along with the tra-

ditional features to achieve better performance. They generally add no computational bulk and are very portable. They perform equally with both neural network based models and standard log-linear models.

Bansal et al. [2014a]; Bansal [2015] describe methods where they use clustering on top of the learned word representations so that these representations can be used in the context of syntactic parsing. In spirit, these representations have similar properties as brown-clusters. The secondary clustering of word representations, while simple, may not be ideal as clustering could lose a lot of information that could have been better used if the representations were directly used.

In a related work, Faruqui et al. [2015b] propose a technique for transforming a set of word vectors to make them more interpretable. They do this by projecting the vectors into a higher-dimensional space, where the vectors are all sparse. Words that are semantically related share nonzero components of these vectors. However, they only test these embeddings on a battery of simple tasks — especially related to word-similarity.

2.5 SUMMARY

We have reviewed the relevant background for lexical representation learning in the field of NLP. We surveyed various techniques of learning word representations and applications of these representations in many NLP tasks.

In Section 2.4, we list some of the challenges in utilizing these word representations, especially for different NLP tasks.

In this context, we propose learning task-specific representations as described in Chapter 3, where we introduce a method that uses a bilinear model to task-specify representations for bilexical prediction tasks. We expand on applications of lexical representations and focus on prepositional phrase attachment ambiguity problem in Chapter 4. We investigate the use of word representations for vocabulary expansion in Chapter 5, and finally, we present techniques for mapping representational spaces in Chapter 6.

3

TAILORING WORD EMBEDDINGS FOR BILEXICAL PREDICTIONS

In this chapter, we investigate the problem of inducing lexical embeddings that are tailored for a particular bilexical relation. We present an approach that takes an existing lexical representation and compresses it such that the resulting word embeddings are good predictors for a target bilexical relation. We study the efficacy of the word embeddings on a battery of tasks on multiple linguistic bilexical relations. We observe that the task-specific embeddings can benefit the quality of the representations, i.e., it learns the necessary task-specific properties. We also show that compressed embeddings can be computationally efficient in lexical prediction tasks.

The chapter starts with the introduction of bilexical operators. In the following section we describe the low-rank bilinear embeddings. We conduct extensive experiments with different word embeddings to show the efficacy of our method.

This work is a part of our published work Madhyastha et al. [2014] and Madhyastha et al. [2015].

3.1 BILEXICAL OPERATORS

Consider learning a model that predicts the probability that an adjective modifies a noun in a sentence. In this case, we would like the bilexical operator to capture the fact that some adjectives are more compatible with some nouns than others. For example, a bilexical operator should predict that the adjective *electronic* has a high probability of modifying the noun *device* but little probability of modifying the noun *case*.

Bilexical operators can be useful for multiple NLP applications. For example, they can be used to reduce ambiguity in a parsing task. Consider the following sentence extracted from a weblog: *Vinyl can be applied to electronic devices and cases* as shown in Figure 3.1. If we

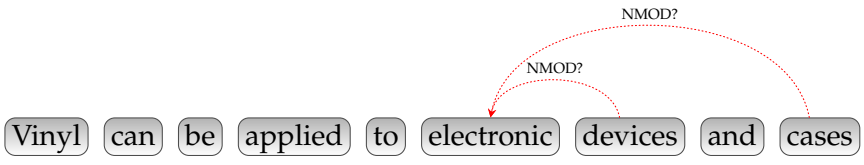


Figure 3.1: The sentence illustrates the ambiguity in deciding the correct attachment of the bilexical relation.

want to predict the dependency structure of this sentence we need to make several decisions. In particular, the parser would need to decide (1) Does *electronic* modify *devices*? (2) Does *electronic* modify *cases*? (3) Does *electronic* modify *both devices* and *cases*? Now imagine that in the corpus used to train the parser none of these nouns have been observed, then it is unlikely that these attachments can be resolved correctly. However, if an accurate noun-adjective bilexical operator were available most of the uncertainty could be resolved. This is because a good bilexical operator would give high probability to the pairs *electronic-device* and low probability to the pair *electronic-case*.

The simplest way of inducing a bilexical operator is to learn it from a training corpus. That is, assuming that we are given some data annotated with a linguistic relation between a modifier and a head (e.g., adjective and noun) we can simply build a maximum likelihood estimator for $\Pr(m|h)$ by counting the occurrences of modifiers and heads under the target relation. For example, we could consider learning bilexical operators from sentences annotated with dependency structures. Clearly, this model can not generalize to head words not present in the training data.

The main limitation of this approach is that the learned bilexical operator can only be evaluated over heads that are present in the supervised corpus but it can not provide information about unknown heads.

To mitigate this we could consider bilexical operators that can exploit lexical embeddings, such as a distributional vector-space representation of words. In this case, we assume that for every word we can compute an n-dimensional vector space representation $\phi(w) \rightarrow \mathbb{R}^n$ as we have presented in Chapter 2. This representation, as we have seen earlier, typically captures distributional features of the context in which the lexical item can occur. The key point is that we do not need a supervised corpus to compute the representation. All we need is a large textual corpus to compute the relevant statistics. Once we have the representation we can exploit operations in the induced vector space to define lexical compatibility operators. For example, we could define a bilexical operator as:

$$\Pr(m \mid h) = \frac{\exp\left\{\langle \phi(m), \phi(h)\rangle\right\}}{\sum_{m'} \exp\left\{\langle \phi(m'), \phi(h)\rangle\right\}} \tag{3.1}$$

where $\langle \phi(x), \phi(y)\rangle$ denotes the inner-product. Alternatively, given an initial high-dimensional distributional representation computed from a large textual corpus we could first induce a projection to a lower k dimensional space by performing truncated singular value decomposition. The idea is that the lower dimensional representation will be more efficient and it will better capture the relevant dimensions of

the distributional representation. The bilexical operator would then take the form of:

$$\Pr(m|h) = \frac{\exp\left\{\langle U\phi(m), U\phi(h)\rangle\right\}}{\sum_{m'} \exp\left\{\langle U\phi(m'), U\phi(h)\rangle\right\}} \tag{3.2}$$

where $U \in \mathbb{R}^{k \times n}$ is the projection matrix obtained via SVD.

The advantage of this approach is that as long as we can estimate the distribution of contexts of words we can compute the value of the bilexical operator. However, this approach has a clear limitation: to design a bilexical operator for a target linguistic relation we must design the appropriate distributional representation. Moreover, there is no clear way of exploiting a supervised training corpus.

An ideal lexical representation should compress the space of lexical words while retaining the essential properties of words in order to make predictions that correctly generalize across words. The typical approach is to first induce a lexical representation in a task-agnostic setting and then use it in different tasks as features. A different approach is to learn a lexical representation tailored for a certain task. In this chapter, we explore the second approach, and employ a formulation to induce task-specific word embeddings. This method departs from a given lexical vector space, and compresses it such that the resulting word embeddings are good predictors for a given lexical relation.

Given the complexity of lexical relations, one expects that the properties of words that are relevant for some lexical relation are different for another relation. This might affect the quality of an embedding, both in terms of its predictive power and the compression it obtains. If we employ a task agnostic low-dimensional embedding, will it retain all the important lexical properties for any relation? And, given a fixed relation, can we further compress an existing word representation?

In this work, we present experiments along these lines that confirm that task-specific embeddings can benefit both quality and computational efficiency of lexicalized predictive models.We test the proposed algorithm on several linguistic relations and show that it can predict modifiers for unknown words more accurately than the unsupervised approach. Furthermore, we compare different types of regularizers for the bilexical operator W, and observe that indeed the low-rank regularizer results in the most efficient technique at prediction time.

3.2 BILINEAR MODELS FOR BILEXICAL PREDICTIONS

In this section, we describe our formulation of bilinear models for bilexical relation. In essence, we combine both the supervised and distributional approaches and present a learning algorithm for inducing bilexical operators from a combination of supervised and unsuper-

vised training data. The main idea is to define bilexical operators using bilinear forms over distributional representations: $\phi(x)^\top W \phi(y)$, where $W \in \mathbb{R}^{n \times n}$ is a matrix of parameters. We can then train our model on the supervised training corpus via conditional maximum-likelihood estimation. To induce a low-dimensional representation, we first observe that the implicit dimensionality of the bilinear form is given by the rank of W. In practice controlling the rank of W can result in important computational savings in cases where one evaluates a target word x against a large number of candidate words y: this is because we can project the representations $\phi(x)$ and $\phi(y)$ down to the low-dimensional space where evaluating the function is simply an inner-product. This setting is in fact usual, for example for lexical retrieval applications (e.g., given a noun, sort all adjectives in the vocabulary according to their compatibility), or for parsing (where one typically evaluates the compatibility between all pairs of words in a sentence).

Consequently, with these ideas, we propose to regularize the maximum-likelihood estimation using a nuclear norm regularizer that serves as a convex relaxation to the rank function. To minimize the regularized objective we make use of an efficient iterative proximal method that involves computing the gradient of the function and performing singular value decompositions.

We first proceed with the definition and some necessary notations. And then describe the formulation of the problem. Lastly, we describe the learning setting.

3.2.1 *Definitions*

Let $\mathcal{V}$ be a vocabulary, and let $x \in \mathcal{V}$ denote a word. Let $\mathcal{H} \subseteq \mathcal{V}$ be a set of head words, and $\mathcal{M} \subseteq \mathcal{V}$ be a set of modifier words. In the noun-adjective relation example, $\mathcal{H}$ is the set of nouns and $\mathcal{M}$ is the set of adjectives.

3.2.2 *Task*

The task is as follows:

We are given a training set of l tuples $\mathcal{D} = \{(m, h)^1, \ldots, (m, h)^l\}$, where $m \in \mathcal{M}$ and $h \in \mathcal{H}$ and we want to learn a model of the conditional distribution $\Pr(m \mid h)$. We want this model to perform well on all head-modifier pairs. In particular, we will test the performance of the model on heads that do not appear in $\mathcal{D}$.

We assume that we are given access to a distributional representation function $\phi : \mathcal{V} \to \mathbb{R}^n$, where $\phi(x)$ is the n-dimensional representation of x. Typically, as we have seen in Chapter 2, this function is computed from an unsupervised corpus [Turney and Pantel, 2010;

Mikolov et al., 2013d; Collobert et al., 2011b]. We use $\phi(x)_{[i]}$ to refer to the i-th coordinate of the vector.

3.2.3 *Bilinear Model*

Our model makes use of the bilinear form $W : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, where $W \in \mathbb{R}^{n \times n}$, and evaluates as $\phi(m)^\top W \phi(h)$. We formulate the bilinear model that, given a query word q, computes a conditional distribution over candidate words c.The models take the following form:

$$\Pr(m \mid h) = \frac{\exp\left\{\phi(m)^\top W \phi(h)\right\}}{\sum_{m' \in \mathcal{M}} \exp\left\{\phi(m')^\top W \phi(h)\right\}} \tag{3.3}$$

The learning problem essentially is to obtain ϕ and W from data, and we approach it in a semi-supervised fashion. We expand on the learning problem in future sections. There exist many approaches to learn ϕ from unlabeled data, we experiment with two approaches: (a) a simple distributional approach where we represent words with a bag-of-words of contextual words; and (b) the skip-gram model by [Mikolov et al., 2013d].

To learn W we assume access to labeled data in the form pairs of compatible examples, i.e. $\mathcal{D} = \{(q, c)^1, \ldots, (q, c)^l\}$, where $q \in \mathcal{H}$ and $c \in \mathcal{M}$. The goal is to be able to predict query-candidate pairs that are unseen during training. Recall that we model relations between words without context. Thus the lexical representation ϕ is essential to generalize to pairs involving unseen words.

Before moving to the next section, let us note that the unsupervised SVD model in Eq. (3.2) is also a bilinear model as defined here. This can be seen if we set $W = UU^\top$, which is a bilinear form of rank k. The key difference is in the way W is learned using supervision.

3.2.4 *Relation to Linear Models*

Note that the above model is nothing more than a conditional log-linear model defined over n^2 features as:

$$f_{i,j}(m, h) = \phi(m)_{[i]} \phi(h)_{[j]} \tag{3.4}$$

Using this as features, we can now write the probabilistic formulation as:

$$\Pr(m|h) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{i,j}(m, h) w_{i,j} \tag{3.5}$$

where, $\mathbf{f}$ and $\mathbf{w}$ are n-dimensional feature vector for (m, h) in an extended feature space and weight vector respectively.

This shows that bilinear models are basically linear models with an extended features space. This allows us, theoretically, to re-utilize all the machinery designed for linear models.

The reason why it is useful to regard W as a matrix will become evident in the next section.

3.2.5 *Learning Low-rank Bilexical Operators*

MOTIVATION: In this section, we begin by motivating bilinear models with low-rank constraint. We can observe that the bilinear form computes a weighted inner product in some space, this is written as:

$$\boxed{\phi(m)^\top W \phi(h)}$$

Rewriting in vectorial notation, we get:

$$\underbrace{\begin{bmatrix}\mathbf{m}_1 & \mathbf{m}_2 & \cdots & \cdots & \cdots & \mathbf{m}_n\end{bmatrix}}_{\phi(m)^\top} \left.\begin{bmatrix} \mathbf{w}_{11} & w_{12} & \cdots & \cdots & \cdots & \mathbf{w}_{1n} \\ \mathbf{w}_{21} & \mathbf{w}_{22} & \cdots & \cdots & \cdots & \mathbf{w}_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{w}_{n1} & \mathbf{w}_{n2} & \cdots & \cdots & \cdots & \mathbf{w}_{nn} \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \vdots \\ \vdots \\ \vdots \\ \mathbf{h}_n \end{bmatrix} \right\} \phi(h)$$

Consider the singular value decomposition of W, in other words factorizing W:

$$\boxed{W = U\Sigma V}$$

This follows:

$$\underbrace{\begin{bmatrix}\mathbf{m}_1 & \mathbf{m}_2 & \cdots & \cdots & \mathbf{m}_n\end{bmatrix}}_{\phi(m)^\top} \underbrace{\left(\underbrace{\begin{bmatrix} \mathbf{u}_{11} & \cdots & \mathbf{u}_{1k} \\ \mathbf{u}_{21} & \cdots & \mathbf{w}_{2k} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \mathbf{u}_{n1} & \cdots & \mathbf{u}_{nn} \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_k \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} \mathbf{v}_{11} & \cdots & \cdots & \mathbf{v}_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{v}_{k1} & \cdots & \cdots & \mathbf{v}_{kn} \end{bmatrix}}_{V^\top} \right)}_{\text{SVD}(W) = U\Sigma V^\top} \left.\begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \vdots \\ \vdots \\ \mathbf{h}_n \end{bmatrix} \right\} \phi(h)$$

Notice, Σ matrix actually contains only non-zero values on the diagnonal. The rest of the matrix is zero. Now, we can write the bilinear form as:

$$\boxed{[\phi(m)^\top U]\ \Sigma\ [V\phi(h)]}$$

Assuming, the matrix W has a rank $k < n$, then:

$$\underbrace{\left(\begin{bmatrix}\mathbf{m}_1 & \mathbf{m}_2 & \cdots & \cdots & \mathbf{m}_n\end{bmatrix}\begin{bmatrix}\mathbf{u}_{11} & \cdots & \mathbf{u}_{1k}\\ \mathbf{u}_{21} & \cdots & \mathbf{w}_{2k}\\ \vdots & \vdots & \vdots\\ \vdots & \vdots & \vdots\\ \mathbf{u}_{n1} & \cdots & \mathbf{u}_{nn}\end{bmatrix}\right)}_{\phi(m)^\top U}\underbrace{\begin{bmatrix}\sigma_1 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \sigma_k\end{bmatrix}}_{\Sigma}\underbrace{\left(\begin{bmatrix}\mathbf{v}_{11} & \cdots & \cdots & \mathbf{v}_{1n}\\ \vdots & \vdots & \vdots & \vdots\\ \mathbf{v}_{k1} & \cdots & \cdots & \mathbf{v}_{kn}\end{bmatrix}\begin{bmatrix}\mathbf{h}_1\\ \mathbf{h}_2\\ \vdots\\ \vdots\\ \mathbf{h}_n\end{bmatrix}\right)}_{V^\top\phi(h)}$$

Further, we can regard $\tilde{m} = \phi(m)^\top U$ as a projection of m and $\tilde{h} = V\phi(h)$ as a projection of h.

Then the bilinear form can be written as:

$$\sum_{i=1}^{n} \Sigma_{[i,i]}\ \tilde{m}_{[i]}\ \tilde{h}_{[i]}$$

This implies that we are essentially re-embedding the $n-$dimensional vectors to a compact $k-$dimensional vector space, which is tailored for a given task and the prediction essentially is computing the inner-product in the $k-$dimensional space.

TRAINING: Given a training set $\mathcal{D}$ and a feature function $\phi(x)$ we can do standard conditional max-likelihood optimization and minimize the negative of the log-likelihood function, $\log \Pr(\mathcal{D})$:

$$\sum_{(m,h)\in\mathcal{D}} \phi(m)^\top W\phi(h) - \log \sum_{m'\in\mathcal{M}} \exp\left\{\phi(m')^\top W\phi(h)\right\} \quad (3.6)$$

The rank of W defines the dimensionality of the induced space. It is easy to see that if W has rank k it can be factorized as $U\Sigma V$ where $U \in \mathbb{R}^{n\times k}$ and $V \in \mathbb{R}^{k\times n}$.

Since the rank of W determines the dimensionality of the induced space, it would be reasonable to add a rank minimization penalty in the objective in (3.6). Unfortunately, this would lead to a non-convex regularized objective. Instead, we propose to use as a regularizer a convex relaxation of the rank function, the nuclear norm $\|W\|_*$ (the ℓ_1 norm of the singular values of W [Srebro et al., 2004]). Putting it all together, the learning algorithm minimizes:

$$\sum_{(m,h)\in\mathcal{D}} -\log \Pr(m \mid h)) + \lambda\|W\|_* \quad (3.7)$$

Here λ is a constant that controls the trade-off between fitting the data and the complexity of the model. This objective is clearly convex since both the objective and the regularizer are convex. To minimize it we use the proximal gradient algorithm which is described next.

A PROXIMAL ALGORITHM FOR BILEXICAL OPERATORS We now describe the learning algorithm that we use to induce the bilexical operators from training data. We are interested in minimizing the objective (3.7), or in fact a more general version where we can replace the regularizer $\|W\|_*$ by standard ℓ_1 or ℓ_2 penalties. For any convex regularizer $r(W)$ (namely ℓ_1, ℓ_2 or the nuclear norm) the objective in (3.7) is convex. Our learning algorithm is based on a simple optimization scheme known as *forward-backward splitting (FOBOS)* [Duchi and Singer, 2009].

This algorithm has convergence rates in the order of $1/\epsilon^2$, which we found sufficiently fast 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.

while *iteration* $<$ *MaxIteration* **do**
  $W_{t+0.5} = W_t - \eta_t g(W_t)$; `// gradient of neg log-likelihood`
  `/* adding regularization penalty:` $W_{t+1} = \text{argmin}_W \|W_{t+0.5} - W\|_2^2 + \eta_t \lambda r(W)$ `*/`
  `/* we use proximal operator */`
  if ℓ_1 *regularizer* **then**
    $W_{t+1}(i,j) = \text{sign}(W_{t+0.5}(i,j)) \cdot \max(|W_{t+0.5}(i,j)| - \eta_t\lambda, 0)$;
    `// Basic thresholding operation`
  else if ℓ_2 *regularizer* **then**
    $W_{t+1} = \frac{1}{1+\eta_t\lambda} W_{t+0.5}$; `// Basic scaling operation`
  else if ℓ_* *regularizer* **then**
    $W_{t+0.5} = U\Sigma V^\top$;
    $\bar{\sigma}_i = \max(\sigma_i - \eta_t\lambda, 0)$; `//` σ_i `= the i-th element on` Σ
    $W_{t+1} = U\bar{\Sigma}V^\top$;
end

Algorithmus 1 : Proximal Algorithm for Bilexical Operators

The FOBOS algorithm works as follows. In a series of iterations $t = 1 \dots T$ compute parameter matrices W_t as follows:

(a) Compute the gradient of the negative log-likelihood, and update the parameters

$$W_{t+0.5} = W_t - \eta_t g(W_t)$$

where $\eta_t = \frac{c}{\sqrt{t}}$ is a step size and $g(W_t)$ is the gradient of the loss at W_t.

(b) We now wish to update $W_{t+0.5}$ to take into account the regularization penalty $r(W)$. That is, we are interested in solving:

$$W_{t+1} = \underset{W}{\text{argmin}} \|W_{t+0.5} - W\|_2^2 + \eta_t \lambda r(W)$$

Principally, in FOBOS, given a particular regularizer, this step is solved using the *proximal operator* associated with the regularizer. Specifically:

- For ℓ_1 it is a simple thresholding:

$$W_{t+1}(i,j) = \text{sign}(W_{t+0.5}(i,j)) \cdot \max(\left|W_{t+0.5}(i,j)\right| - \eta_t\lambda, 0)$$

- For ℓ_2 it is a simple scaling:

$$W_{t+1} = \frac{1}{1+\eta_t\lambda} W_{t+0.5}$$

- For nuclear-norm, perform SVD thresholding. Compute the SVD to write $W_{t+0.5} = USV^\top$ with S a diagonal matrix and U, V orthogonal matrices. Denote by σ_i the i-th element on the diagonal of S. Define a new matrix $\bar{S}$ with diagonal elements $\bar{\sigma}_i = \max(\sigma_i - \eta_t\lambda, 0)$. Then set

$$W_{t+1} = U\bar{S}V^\top$$

Optimizing a bilinear model using nuclear-norm regularization involves an extra, relatively small, cost of performing SVD of W at each iteration. The optimization parameters of the method are the regularization constant λ, the step size constant c and the number of iterations T. In our experiments we ran a range of λ and c values for 200 iterations, and used a validation set to pick the best configuration.

3.3 RELEVANT RELATED WORK

There have been many strands of related research in the past few years that have a similar objective. Some of the earliest work especially in Blei and McAuliffe [2008] implement sLDA where they use topic variables as the supervision to learn a log-linear model.

Bai et al. [2010] use a technique similar ours, using bilinear forms with low-rank constraints. In their case, they explicitly look for a low-rank factorization of the matrix, making their optimization non-convex. As far as we know, ours is the first convex formulation. They apply the method to document ranking and thus optimize a max-margin ranking loss. In our application to bilexical models, we perform conditional max-likelihood estimation. Chechik et al. [2010] also learned bilinear operators using max-margin techniques, with pairwise similarity as supervision, but they did not consider low-rank constraints.

One related area where bilinear operators are used to inducing embeddings is distance metric learning. Weinberger and Saul [2009] used large-margin nearest neighbor methods to learn a non-sparse embedding, but these are computationally intensive and might not be suitable for large-scale tasks in NLP.

3.4 EXPERIMENTS, ANALYSIS AND RESULTS

3.4.1 *Experiments on Syntactic Relations with Distributional Representation*

We conducted a set of experiments to test the ability of our algorithm to learn bilexical operators for several linguistic relations. As supervised training data we use the gold standard dependencies of the WSJ training section of the Penn Treebank [Marcus et al., 1993b]. We consider the following relations:

- Noun-Adjective: we model the distribution of adjectives given a noun; and a separate distribution of nouns given an adjective.
- Verb-Object: we model the distribution of object nouns given a verb; and a separate distribution of verbs given an object.

The distributional representation $\phi(x)$ was computed using the BLLIP corpus [Charniak et al., 2000]. We compute a bag-of-words representation for the context of each lexical item, that is $\phi(w)_{[i]}$ corresponds to the frequency of word i appearning in the context of w.

We experiment with several settings while varying different parameters. We use a context window of size 10 and restrict our bag-of-words vocabulary to contain only the 2,000 most frequent words present in the corpus. Vectors were normalized.

To test the performance of our algorithm for each relation we partition the set of heads into a training and a test set, 60% of the heads are used for training, 10% of the heads are used for validation and 30% of the heads are used for testing. Then, we consider all observed modifiers in the data to form a vocabulary of modifier words. The goal of this task is to learn conditional distribution over all these modifiers given a head word without context. In our experiments, the number of modifiers per relation ranges from 2,500 to 7,500 words. For each head word, we create a list of *compatible* modifiers from the annotated data, by taking all modifiers that occur at least once with the head. Hence, for each head, the set of all modifiers is partitioned into compatible and non-compatible. For testing, we measure a *pairwise accuracy*, the percentage of compatible/non-compatible pairs of modifiers where the former obtains higher probability. We stress that none of the test head words has been observed in training, while the list of modifiers is the same for training, validation, and testing.

We compare the performance of the bilexical model trained with nuclear norm regularization (ℓ_*) with other regularization penalties (ℓ_1 and ℓ_2). We also compare these supervised methods with an unsupervised model: a low-dimensional SVD model as in Eq. (3.2), which corresponds to an inner product as in Eq. (3.1) when all dimensions are considered.

To report performance, we measure pairwise accuracy with respect to the capacity of the model in terms of the number of active parameters. To measure the capacity of a model we consider the number of double operations that are needed to compute, given a head, the scores for all modifiers in the vocabulary (we exclude the exponentiations and normalization needed to compute the distribution of modifiers given a head, since this is a constant cost for all the models we compare and is not needed if we only want to rank modifiers). Recall that the dimension of $\phi(x)$ is n, and assume that there are m total modifiers in the vocabulary. In our experiments $n = 2,000$ and m ranges from $2,500$,to $7,500$. The correspondences with operations are:

- Assume that the ℓ_1 and ℓ_2 models have k non-zero weights in W. Then the number of operations to compute a distribution is km.
- Assume that the ℓ_* and the unsupervised models have rank k. We assume that the modifier vectors are already projected down to k dimensions. For a new head, one needs to project it and perform m inner products, hence the number of operations is $kn + km$.

We note that if we set W to be the identity matrix our model scores are inner products between the query-candidate embeddings, a common approach to evaluating semantic similarity in unsupervised distributional approaches.

In general, we can compute a low-dimensional projection of ϕ down to k dimensions, using SVD, and perform the inner product in the projected space. We refer to this as the unsupervised approach, since the projected embeddings do not use the labeled dataset specifying the target relation.

RESULTS AND DISCUSSION: Figure 3.2 shows the performance of models for noun-adjective and verb-object relations, while Figure 3.3 shows plots for prepositional relations.[1] The first observation is that supervised approaches outperform the unsupervised approach. In cases such as noun-adjective relations, the unsupervised approach performs close to the supervised approaches, suggesting that the pure distributional approach can sometimes work. But in most relations, the improvement obtained by using supervision is very large. When comparing the type of regularizer, we see that if the capacity of the model is unrestricted (right part of the curves), all models tend to perform similarly. However, when restricting the size, the nuclear-norm model performs much better.

1 To obtain curves for each model type with respect to a range of the number of operations, we first obtained the best model on validation data and then forced it to have at most k non-zero features or rank k by projecting, for a range of k values.

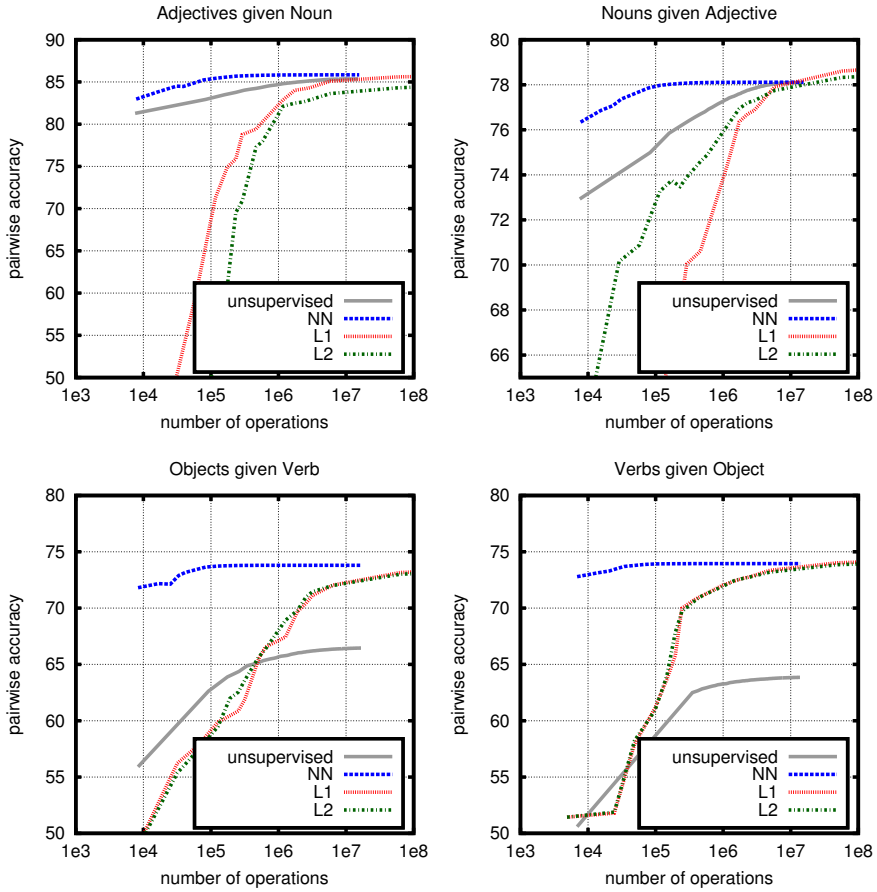


Figure 3.2: Pairwise accuracy with respect to the number of double operations required to compute the distribution over modifiers for a head word. Figures for noun-adjective and verb-object relations, in both directions.

Noun	Predicted Adjectives
president	executive, senior, chief, frank, former, international, marketing, assistant, annual, financial
wife	former, executive, new, financial, own, senior, old, other, deputy, major
shares	annual, due, net, convertible, average, new, high-yield, initial, tax-exempt, subordinated
mortgages	annualized, annual, three-month, one-year, average, six-month, conventional, short-term, higher, lower
month	last, next, fiscal, first, past, latest, early, previous, new, current
problem	new, good, major, tough, bad, big, first, financial, long, federal
holiday	new, major, special, fourth-quarter, joint, quarterly, third-quarter, small, strong, own

Table 3.1: 10 most likely adjectives for some test nouns.

Roughly, 20 hidden dimensions are enough to obtain the most accurate performances (which result in $\sim 140,000$ operations for initial representations of $2,000$ dimensions and $5,000$ modifier candidates). As an example of the type of predictions, Table 3.1 shows the most likely adjectives for some test nouns.

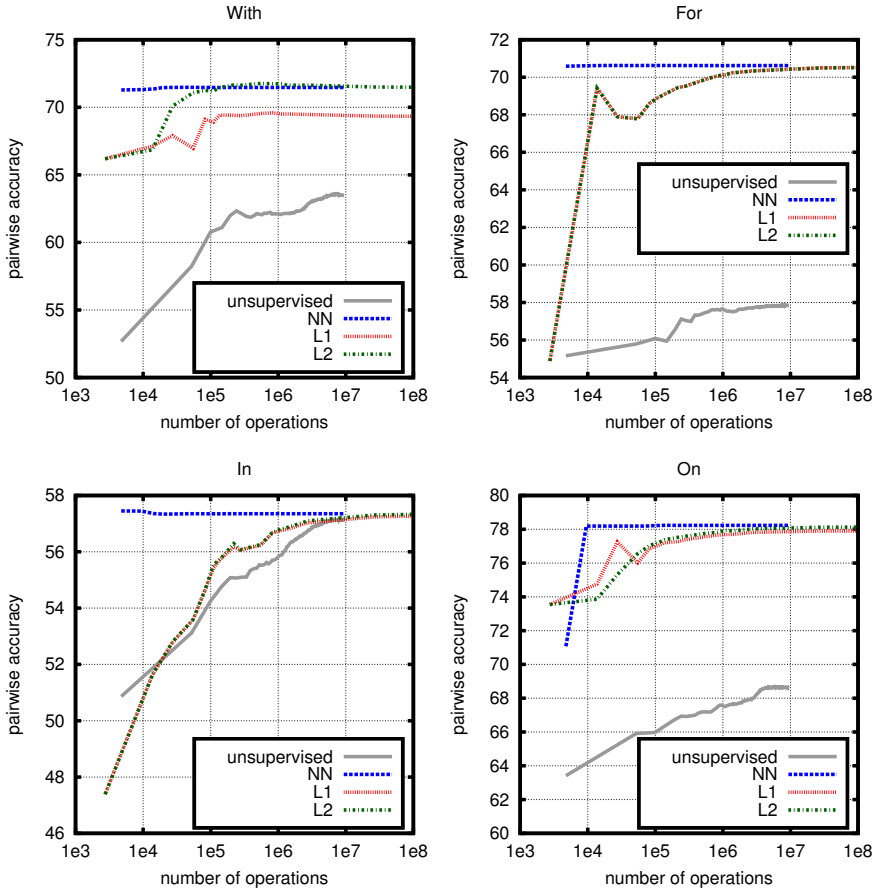


Figure 3.3: Pairwise accuracy with respect to the number of double operations required to compute the distribution over modifiers for a head word. Figures for four prepositional relations: with, for, in, on. The distributions are of verbs and objects above the preposition given the noun below the preposition.

3.4.2 *Experiments with Distributional and Distributed Representations*

We conducted a set of experiments to test the performance of the learning algorithm with respect to the initial lexical representation ϕ, for different configurations of the representation and the learner. We again experiment with six bilexical syntactic relations using the Penn Treebank corpus Marcus et al. [1993b], following the same experimental setting as in the previous section. For a relation between queries and candidate words, such as noun-adjective, we partition the query words into train, development and test queries, thus test pairs are always unseen pairs.

As in the previous section, we measure pairwise accuracy with respect to the efficiency of the model in terms of the number of active parameters. To measure the efficiency of a model we consider the number of double operations that are needed to compute, given a query word, the scores for all candidates in the vocabulary.

We experiment with two types of initial representations ϕ. The first is a simple high-dimensional distributional representation based on contextual bag-of-words (BoW): each word is represented by the bag of words that appear in contextual windows. In our experiments, these were sparse 2,000-dimensional vectors. The second representation are the low-dimensional skip-gram embeddings (SKG) by [Mikolov et al., 2013d], where we used 300 dimensions. In both cases, we induced such representations using the BLIPP corpus Charniak et al. [2000] and using a context window of size 10 for both. Thus the main difference is that the bag-of-words are an uncompressed representation, while the skip-gram embeddings are a neural-net-style compression of the same contextual windows.

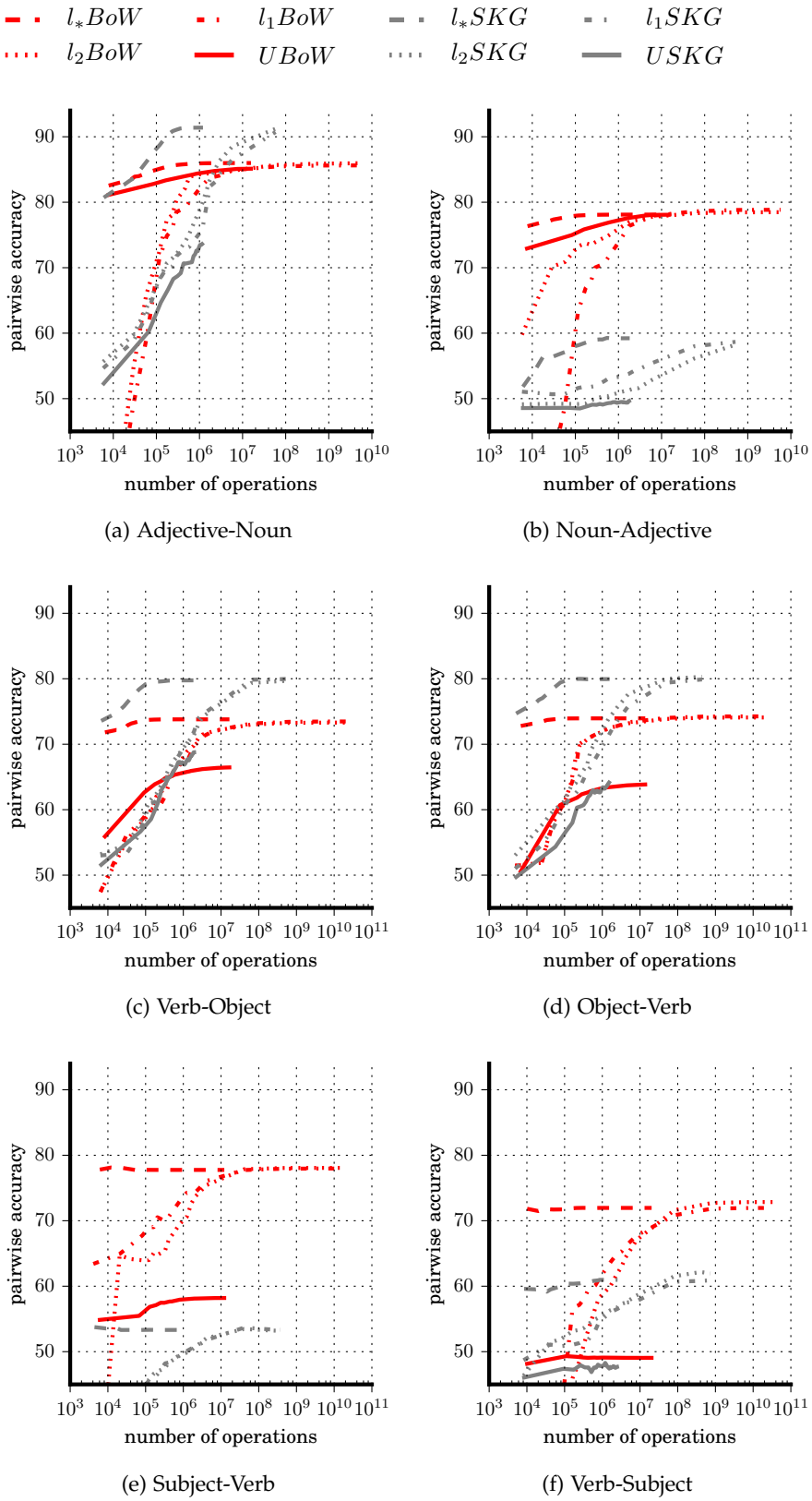
As for the bilexical model, we test it under three regularization schemes, namely ℓ_2, ℓ_1, and $\ell_\star$. For the first two, the efficiency of computing predictions is a function of the non-zero entries in W, while for the latter it is the rank k of W, which defines the dimension of the task-specific embeddings. We also test a baseline unsupervised approach (UNS).

RESULTS AND DISCUSSION: Figure 3.4 shows the performance of models for noun-adjective, verb-object and verb-subject relations (in both directions). In line with the results by observed in the previous section, we observe that the supervised approach in all cases outperforms the unsupervised case, and that the nuclear norm scheme provides the best performance in terms of accuracy and speed: other regularizers can obtain similar accuracies, but low-rank constraints during learning favor very low dimensional embeddings that are highly predictive.

In terms of starting with bag-of-words vectors or skip-gram embeddings, in three relations the former is clearly better, while in the other three relations the latter is clearly better. We conclude that task-agnostic embeddings do identify useful relevant properties of words, but at the same time, not all necessary properties are retained. In all cases, the nuclear norm regularizer successfully compresses the initial representation, even for the embeddings which are already low-dimensional.

Table 3.2 presents the best result for each relation, initial representation and regularization scheme. Plus, for the $\ell_\star$ regularizer we present results at three different ranks, namely 5, 10 or the rank that obtains the best result for each relation. These highly compressed embeddings perform nearly as good as the best performing model for each relation.

Table 3.3 shows a set of query nouns, and two sets of neighbor query nouns, using the embeddings for two different relations to compute the two sets. We can see that, by changing the target relation, the set of close words changes. This suggests that words have



(a) Adjective-Noun

(b) Noun-Adjective

(c) Verb-Object

(d) Object-Verb

(e) Subject-Verb

(f) Verb-Subject

Figure 3.4: Pairwise accuracy v/s no. of double operations to compute the distribution over candidate words for a query word. Plots are for noun-adjective, verb-object and verb-subject relations, in both directions. The red curves use distributional representation based on bag-of-words (BoW) and the grey curves use the embeddings of the skip-gram model (SKG).

			$\ell_\star$				
Rel	*Type*	*UNS*	*best* k	k = 5	k = 10	ℓ_2	ℓ_1
Adj-Noun	BoW	85.12	85.99 (80)	83.99	84.74	85.96	85.63
	SKG	73.61	91.40 (300)	83.70	86.27	91.22	90.72
Obj-Verb	BoW	63.85	78.11 (200)	73.17	73.64	74.08	73.95
	SKG	64.15	79.98 (50)	75.45	78.37	80.30	79.89
Subj-Verb	BoW	58.20	78.13 (2)	71.71	71.73	78.07	77.97
	SKG	49.65	59.28 (90)	53.31	53.32	58.24	58.67
Noun-Adj	BoW	78.09	78.11 (70)	77.48	77.85	78.48	78.85
	SKG	49.65	59.28 (50)	56.42	57.19	58.24	58.67
Verb-Obj	BoW	66.46	73.90 (40)	73.70	73.88	73.30	73.48
	SKG	64.15	79.99 (30)	77.05	78.60	80.29	79.89
Verb-Subj	BoW	49.32	71.97 (30)	71.71	71.23	72.85	71.95
	SKG	32.34	53.75 (2)	53.32	53.32	53.47	53.68

Table 3.2: Pairwise accuracies for the six relations using the unsupervised, $\ell_\star$, ℓ_2 and ℓ_1 models, using either a distributional bag-of-words representation (BoW) or the skip-gram embeddings (SKG) as initial representation. For $\ell_\star$ we show results for the rank that gives best accuracy (with the optimal rank in parenthesis), as well as for ranks k = 5 and 10.

a wide range of different behaviors, and different relations might exploit lexical properties that are specific to the relation.

3.5 SUMMARY

We have presented a model for learning bilexical operators that can leverage both supervised and unsupervised data. The model is based on exploiting bilinear forms over distributional representations. The learning algorithm induces a low-dimensional representation of the lexical space by imposing low-rank constraints on the parameters of the bilinear form. By means of supervision, our model induces two low-dimensional lexical embeddings, one on each side of the bilexical linguistic relation, and computations can be expressed as an inner-product between the two embeddings. This factorized form of the model can have great computational advantages: in many applications, one needs to evaluate the function multiple times for a fixed set of lexical items, for example in dependency parsing. Hence, one can first project the lexical items to their embeddings, and then compute all pairwise scores as inner-products. In experiments, we have shown

Query	noun-adjective	object-verb
city	province, area, island, township, freeways	residents, towns, marchers, streets, mayor
securities	bonds, mortgage, issuers, debt, loans	bonds, memberships, equities, certificates, syndicate
board	committee directors, commission, nominees, refusal	slate, membership, committee, appointment, stockholder
debt	loan, loans, debts, financing, mortgage	reinvestment, indebtedness, expenditures, outlay, repayment
law	laws, constitution, code, legislation, immigration	laws, ordinance, decree, statutes, state
director	assistant, editor, treasurer, postmaster, chairman	firm, consultant, president, manager, leader

Table 3.3: Example query words and 5 highest-ranked candidate words for two different bilexical relations: noun-adjective and object-verb.

that the embeddings we obtain in a number of linguistic relations can be modeled with a few hidden dimensions.

As future work, we would like to apply the low-rank approach to other model forms that can employ lexical embeddings, specially when supervision is available. For example, dependency parsing models, or models of predicate-argument structures representing semantic roles, exploit bilexical relations. In these applications, being able to generalize to word *pairs* that are not observed during training is essential.

We would also like to study how to combine low-rank bilexical operators, which in essence induce a task-specific representation of words, with other forms of features that capture class or contextual information. One desires that such combinations can preserve the computational advantages behind low-rank embeddings.

We have presented a set of experiments where we compute word embeddings specific to target linguistic relations. We observe that low-rank penalties favor embeddings that are good both in terms of predictive accuracy and efficiency. For example, in certain cases, models using very low-dimensional embeddings perform nearly as good as the best models.

In certain tasks, we have shown that we can refine low-dimensional skip-gram embeddings, making them more compressed while retaining their predictive properties. In other tasks, we have shown that our method can improve over skip-gram models when starting from uncompressed distributional representations. This suggests that skip-gram embeddings do not retain all the necessary information of the original words. This motivates future research that aims at general-purpose embeddings that do retain all necessary properties and can be further compressed in light of specific lexical relations.

4

RESOLVING PREPOSITIONAL PHRASE AMBIGUITY USING WORD-EMBEDDINGS

Inspired from our previous results, we continue to explore our proposal on one of the well defined task in the field of NLP — resolution of Prepositional Phrase Ambiguity. This chapter is also a part of our ongoing submission titled *Prepositional Phrase Attachment over Word Embedding Products* and the germination of the idea is based on our earlier work in Madhyastha et al. [2014].

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. We conduct experiments for a wide range of datasets, task settings and word embeddings. Our results 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. Furthermore, its performance is close to that of models that use additional knowledge sources for semantic information such as WordNet and VerbNet. Finally, our model gives the current best reported performance on an out-of-domain evaluation and also performs competitively with other parsers in the context of PP task.

4.1 INTRODUCTION

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

Consider the examples in Figure 4.1. 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].

In recent years, word embeddings have become a very popular representation for lexical items [Mikolov et al., 2013a; Pennington et al., 2014b]. The idea is that the dimensions of a word embedding capture lexical, syntactic, and semantic features of words –in essence, the

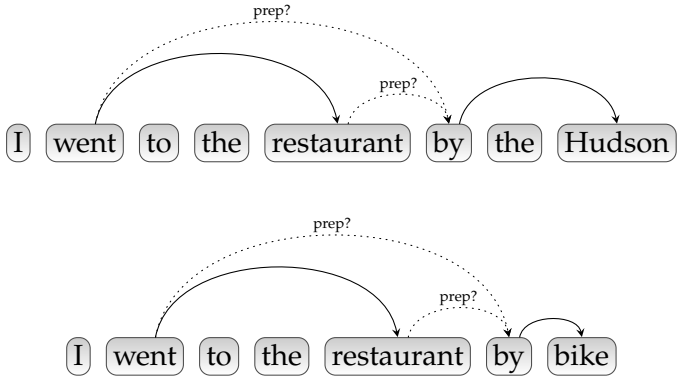


Figure 4.1: PP Attachment Ambiguity

type of information that is exploited in PP attachment systems. Recent work in dependency parsing [Chen and Manning, 2014b; 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 further 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 for 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 seem to suggest that for the 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.

In summary, our contributions are:

- We present a simple multi-linear model for PP attachment that makes use of tensor products of word embeddings, capturing all possible conjunctions of latent embeddings.

- We conduct several experiments comparing the performance of different word embeddings and composition operations for the PP attachment task under different settings. Our results show that tensor products are the best compositional operation and that word embeddings that include syntactic information, such as skip-dep [Bansal et al., 2014b], are significantly better for the task than other popular embeddings.

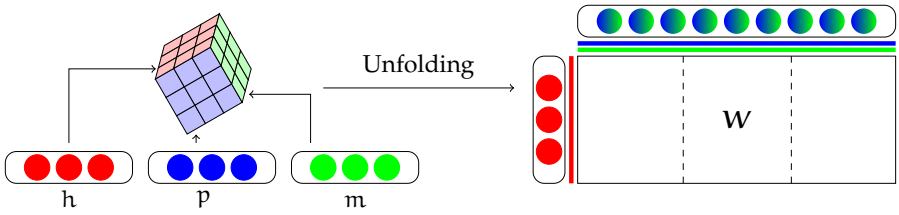


Figure 4.2: The tensor product of word embeddings. Here, h, p, and m are the head, preposition and modifier of the PP attachment structure, represented by their word embeddings. The tensor product forms a cube, which we unfold with respect to the head and the prepositional phrase, resulting in a matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n^2}$.

- When compared to the state-of-the-art models, our results show that a relatively simple multi-linear model that uses only word embeddings can outperform more complex non-linear architectures that exploit the same information. Furthermore, its performance is close to that of models that use additional knowledge sources for semantic information such as WordNet and VerbNet. This seems to suggest that products of word embeddings should be the core feature space of choice to resolve lexical attachment ambiguities.

- For out-of-domain tests, we observe significant improvements just by using word embeddings trained on unlabeled data from the target domains. With these improvements, our tensor products outperform state-of-the-art dependency parsers on PP attachment decisions.

4.2 PP ATTACHMENT

Ratnaparkhi et al. [1994a] 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 $p - m$ 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 $p - m$ prepositional phrase. The binary case corresponds to $H = \{v, o\}$. This generalized setting directly captures the PP attachment problem in the context of dependency parsing, where multiple attachment candidates are considered (e.g., all verbs and nouns of a sentence)

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:

$$\arg\max_{h \in H} f(h, p, m) \quad , \tag{4.1}$$

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

4.3 TENSOR PRODUCTS FOR PP ATTACHMENT

In this section, we present models for PP attachment based on tensor products of word embeddings.

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

The tensor product model for PP attachment is as follows (see also Figure 4.2):

$$f(h, p, m) = \boldsymbol{v}_h^\top \boldsymbol{W} [\boldsymbol{v}_p \otimes \boldsymbol{v}_m] \quad , \tag{4.2}$$

where $\boldsymbol{W} \in \mathbb{R}^{n \times n^2}$ is a matrix of parameters, taking the embedding of the attachment candidate h on the left, and the product of embeddings of the $p - m$ 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 (4.2) 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 al-

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

gorithm imposes low-rank constraints on $\boldsymbol{W}$ (see Section 4.3.2 below), for which the unfolding of the tensor becomes relevant.

4.3.1 *Variations of the Tensor*

We now discuss variations to the above model. In all cases we will write our models as bilinear functions of the following form:

$$f(h, p, m) = \boldsymbol{\alpha}(h)^\top \boldsymbol{W} \boldsymbol{\beta}(p, m) \tag{4.3}$$

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 expressiveness 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 rarer 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 $\boldsymbol{i}_p \in \mathbb{R}^{|\mathcal{P}|}$ 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:

  $$f(h, p, m) = \boldsymbol{v}_h^\top \boldsymbol{W}_p \boldsymbol{v}_m \tag{4.4}$$

  where we have one separate parameter matrix $\boldsymbol{W}_p \in \mathbb{R}^{n \times n}$ per preposition p. This is the simplest model that we first explore.

- **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, ...). In vector form, let $\boldsymbol{\delta}_h \in \mathbb{R}^{|H|}$ 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:

  $$f(h, p, m) = \boldsymbol{v}_h^\top \boldsymbol{W}_{\delta_h} \boldsymbol{v}_m \quad . \tag{4.5}$$

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.

4.3.2 *Low-rank Matrix Learning*

Similar to our previous work, 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, 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 $\boldsymbol{U} \in \mathbb{R}^{n \times k}$ and $\boldsymbol{V} \in \mathbb{R}^{n^2 \times k}$. 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:

$$\arg\max_{\boldsymbol{W}} \text{logistic}(\mathcal{T}, \boldsymbol{W}) + \lambda \|\boldsymbol{W}\|_\star \tag{4.6}$$

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]. A detailed explanation can be seen in 3.2.5

4.4 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.

4.4.1 *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. [1994a] (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 pre-process 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:

- The test set from the RRR dataset, with 3,097 samples from the PTB.
- The New York Times test set (NYT) released by Nakashole and Mitchell [2015]. It contains 293 test samples.
- 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 SPRML shared task data [Seddah et al., 2010], and consists of 40,121 training samples and 3,647 test samples.

4.4.2 *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 configurations are as follows:

(a) **Bag-of-Words Representation**: To explore the complementarity in 4.4.3, we utilized the distributional representation computed using the BLLIP corpus [Charniak et al., 2000]. We computed a

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

bag-of-words representation for the context of each lexical item restricting the bag-of-words vocabulary to contain only 2,000 most frequent words and the vectors were normalized.

(b) **skip-gram** [Mikolov et al., 2013a]: 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] We train word embeddings for each of the following data sources:

- BLLIP [Charniak et al., 2000], with ~1.8 million sentences and ~43 million tokens of Wall Street Journal text (and excludes PTB evaluation sets).
- English Wikipedia[4], with ~13.1 million sentences and ~129 million tokens.
- The New York Times portion of the GigaWord corpus, with ~52 million sentences and ~1,253 million tokens.

(c) **skip-dep** [Bansal et al., 2014b]: 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 BLLIP corpus in the same setting as described in Bansal et al. [2014b], but using TurboParser [Martins et al., 2013] [5] to obtain dependency trees for BLLIP. 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 4.3, 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.[6]

4.4.3 *Experiments on the Binary Attachment Setting*

3 In preliminary experiments we tried a window of 2, which performed worse in our setting. According to Bansal et al. [2014b] with larger context window, words that is topically-related tend to get closer. While with small window size, close words tend to share the same POS tag, which is less relevant for PP attachment because the position in the attachment structure already indicates the POS tag.

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

5 `http://www.cs.cmu.edu/~ark/TurboParser`

6 Throughout this section, whenever we refer to vectors of dimension n, we actually work with vectors of dimension $n+1$.

COMPLEMENTARITY OF LINEAR AND BILINEAR MODELS This section presents a series of experiments using the classic binary setting by Ratnaparkhi et al. [1994a].

We first begin by exploring PP attachment ambiguity as a simple bilexical prediction task per preposition. We start from the formulation of the task as a binary classification problem by Ratnaparkhi et al. [1994b]: given a tuple $x = \langle v, o, p, n \rangle$ consisting of a verb v, noun object o, preposition p and noun n, decide if the prepositional phrase p-n attaches to v ($y = v$) or to o ($y = o$). Ratnaparkhi et al. [1994b] define a linear maximum likelihood model of the form $\Pr(y \mid x) = \exp\{\langle w, f(x, y)\rangle\} * Z(x)^{-1}$, where $f(x, y)$ is a vector of d features, w is a parameter vector in $\mathbb{R}^d$, and $Z(x)$ is the normalizer summing over $y = \{v, o\}$. Here we define a bilexical model of the form that uses a distributional representation(ϕ) per preposition:

$$\Pr(v|\langle v, o, p, n\rangle) = \frac{\exp\{\phi(v)^\top W_v^p \phi(n)\}}{Z(x)}$$

and

$$\Pr(o|\langle v, o, p, n\rangle) = \frac{\exp\{\phi(o)^\top W_o^p \phi(n)\}}{Z(x)}$$

This setting is principally similar to our earlier proposal in Section 4.3.1, where we use an indicator vector for the preposition. The bilinear model is parameterized by two matrices W_v and W_o per preposition, each of which captures the compatibility between nouns below a certain preposition and heads of v or o prepositional relations, respectively. Again $Z(x)$ is the normalizer summing over $y = \{v, o\}$, but now using the bilinear form.

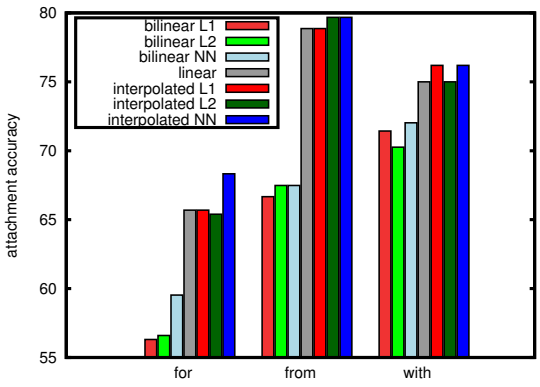


Figure 4.3: Attachment accuracies of linear, bilinear and interpolated models for three prepositions.

We use exclusively the bag-of-words representation. We ran experiments using the data by Ratnaparkhi et al. [1994b]. We trained separate models for different prepositions, focusing on the prepositions

that are more ambiguous: `for`, `from`, `with`. We compare to a linear 'maxent' model following Ratnaparkhi et al. [1994b] that uses the same feature set. Figure 4.3 shows the test results for the linear model, and bilinear models trained with ℓ_1, ℓ_2, ℓ_* (in figure labelled as L1, L2, NN respectively) regularization penalties. However, note that we use sparse distributional embeddings for this specific setting. The results of the bilinear models are significantly below the accuracy of the linear model, suggesting that some of the non-lexical features of the linear model (such as prior weighting of the two classes) might be difficult to capture by the bilinear model over lexical representations. To check if the bilinear model might complement the linear model or just be worse than it, we tested simple combinations based on linear interpolations. For a constant $\lambda \in [0, 1]$ we define:

$$\Pr(y \mid x) = \lambda \Pr_L(y \mid x) + (1 - \lambda) \Pr_B(y \mid x) \quad . \tag{4.7}$$

We search for the best λ on the validation set and report results of combining the linear model with each of the three bilinear models. Results are shown also in Figure 4.3. Interpolation models improve over linear models, though only the improvement for `for` is significant (2.6%). Future work should exploit finer combinations between standard linear features and distributional bilinear forms.

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. 4.2, that resolves the attachment using only a product of word embeddings, and used ℓ_* regularization. Table 4.1 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).[7] 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 The peak performance is for skip-dep using 100 dimensional vectors trained on BLLIP. [8] 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 4.3.1 presents variations that

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

8 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 4.1: 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.

compose the prepositional phrase (i.e. the preposition and modifier vectors) in different ways. We now compare these variants empirically, using skip-dep vectors with $n = 50$ as word embeddings. Table 4.2 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 clearly show that the product model is the best of all, 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.

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	$[\boldsymbol{i}_p \otimes \boldsymbol{v}_m]$	$n \times \lvert\mathcal{P}\rvert * n$	84.36
Product	$[\boldsymbol{v}_p \otimes \boldsymbol{v}_m]$	$n \times n * n$	**85.52**

Table 4.2: Development accuracy for several ways of composing the word embeddings of the prepositional phrase. $\boldsymbol{i}_p \in \mathbb{R}^{|\mathcal{P}|}$ denotes an indicator vector for preposition p, where $\mathcal{P}$ is the set of prepositions.

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 4.3, which lists representative and

		Test Accuracy		
Method	Word Embedding	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 4.3: Accuracy results over the RRR, NYT and WIKI test sets. (*) indicates that the system uses additional semantic features.

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 obtained 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.

4.4.4 *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 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. 4.5. Without it, the performance was much worse (possibly because in this data, a large number of samples attach to the first or the second candidate in the list —about 93% of cases on the English data).

	Test Accuracy	
System	Arabic	English
Tensor product ($n=50$, ℓ_*)	-	88.3
Tensor product ($n=50$, ℓ_2)	-	87.8
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 4.4: Test accuracy for PP attachment with multiple head candidates.

Table 4.4 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. In any case, one characteristic of low-rank regularization is the inherent compression of the tensor. Figur 4.4 plots accuracy versus rank for the tensor working with 50-dimensional embeddings composed with positional information (min dimension is 357): with rank 50 the model obtains 88% of accuracy. 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.

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.

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, 2012a], which has annotated evaluation sets for five domains, and extracted PP-attachment tuples using the procedure described by Belinkov et al.

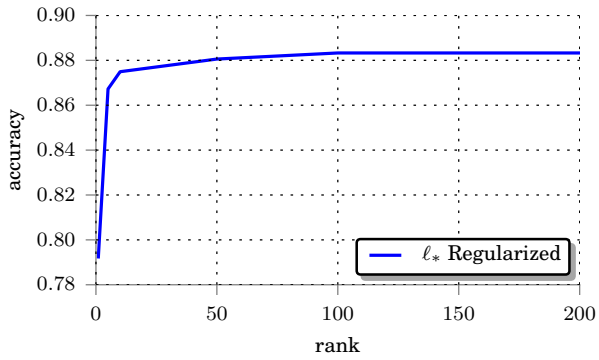


Figure 4.4: Accuracy versus rank of the tensor model on the English data by Belinkov et al. [2014]. The tensor model uses 50-dimensional vectors composed with head position, and has size $357 \times 2,500$.

	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 4.5: 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 $n = 100$.

[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.[9] 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, 2014b], which uses word embeddings but not as products.[10] The other is TurboParser [Martins et al., 2013][11] 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

9 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.

10 We used Stanford CoreNLP 3.7.0. We could not determine the characteristics of the embeddings in the model.

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

from the parse tree. [12] We also retrained Stanford parser with default parameters using our skip-dep embeddings trained over both BLLIP and the unlabelled data of the Web Treebank. However, there were no noticeable change in the performance of the parser on the PP attachments. 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.[13]

Table 4.5 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 over 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.[14] 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.

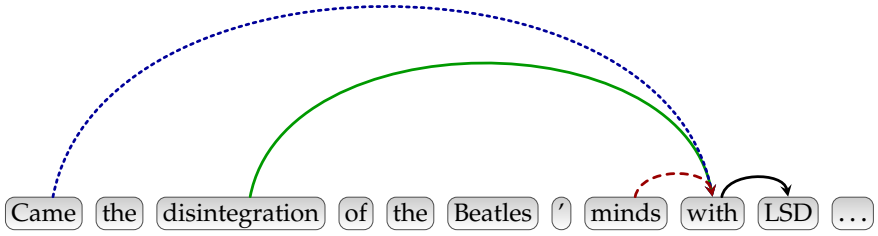
ERROR ANALYSIS. To further understand the performance of the tensor products and parsers on WTB development set, we consider PP attachment instances where the words are observed less than five times in the training data (1,565 cases out of 4,655). The best tensor product obtains an accuracy of 84.3% (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 4.5 shows two sample sentences from the Web Treebank that illustrate two cases of ambiguities. Sentence (a) is an example of lexical paucity, because of 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

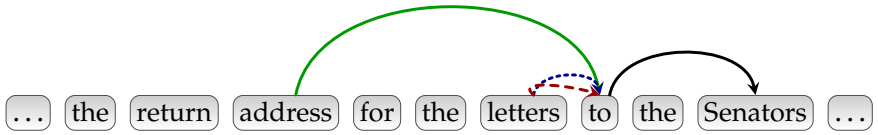
12 We ran all parsing models on correct PoS tags. Thus, these are optimistic performances. This choice rules out cases where the parsers fail because of tagging errors, which would be unfair because our models work on pre-selected head candidate lists which depend on correct PoS tags.

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

14 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 $p < 0.01$.



(a)



(b)

Figure 4.5: 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). Here, (a) The modifier and correct head are unseen in training, and; (b) The correct head is ambiguous.

between parsers and products of embeddings, and these examples suggest combinations of both.

4.5 RELEVANT RELATED WORK

4.5.1 *Resolving PP Attachment Ambiguity*

Several approaches have been proposed for solving the prepositional attachment problem, including maximum likelihood with back-off [Hindle and Rooth, 1993; Collins and Brooks, 1999], and discriminative training [Ratnaparkhi et al., 1994a; 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 make use as word embeddings as the only source of lexical information. Previous work has also explored word representations as extra features [Zhao and Lin, 2004]. In our case, we define a model that exploits all possible conjunctions of the vectors involved in an attachment decision. Our model is, in fact, a generalization of that of Madhyastha et al. [2014], as described in section 4.3.1.

From that work, our application to PP attachment mainly differs in using compact word embeddings as opposed to sparse distributional vectors.

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 of how different types of word embeddings affect the performance of 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., 2010b; Chen and Manning, 2014b; Dyer et al., 2015a] to name a few, or averaging [Socher et al., 2011; Huang et al., 2012b]. Compositions based on the product of embeddings have been explored in tensor models, which we discuss next.

4.5.2 *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 controlling 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. One 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 approach 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; 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 a Tucker form [Lei et al., 2014; Yu et al., 2016]. In this case, for a tensor of d modes, the Tucker form has one projection matrix for each of the modes. 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.

In the context of relation extraction, Singh et al. [2015] did an experimental comparison of different forms of low-rank matrix and tensor learning, showing that they have complementary benefits.

4.6 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 the lexical information only, and are close in performance to methods using richer feature spaces. The accuracies we obtain are particularly good in out-of-domain tests. Since 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.

5

VOCABULARY EXPANSION OVER WORD EMBEDDINGS

In this chapter, we extend our previous conclusions and propose to use a simple log-bilinear softmax based model for vocabulary expansion, such that, given an out of vocabulary source word the model generates a probabilistic list of possible translations in the target language. Our model uses only word-embeddings trained on a significantly large unlabelled corpus and trains over a fairly small, word-to-word bilingual dictionary. We input this probabilistic list into a standard phrase based machine translation system and obtain consistent improvements in translation quality on English-Spanish language pair. Especially, we get an improvement of around 3.9 BLEU points when tested over an out-of-domain testset. This work is a part of our ongoing submission titled *Vocabulary Expansion for Machine Translation by Mapping Embeddings*. The central idea of this work is also used in our recent paper Costa-jussà et al. [2016].

5.1 INTRODUCTION

Distributed word vector representations have been increasingly used in a variety of tasks in the field of NLP and beyond. It has been proven to be useful for many NLP tasks.

This encouraging performance has generated a fair amount of interest in extending to bilingual and multilingual shared embedding approaches. It has also been shown that the quality of word embeddings can be improved by sharing some attributes amongst the languages in both syntactic and semantic tasks. Different kinds of strategies have been proposed to induce bilingual or multilingual embeddings from methods that make use of document level information to sentence level information and word level information. It has been shown recently that the embeddings have different properties with different strategies of inducing information.

A standard approach of learning mappings or inducing cross-lingual information is by using a form of linear regression based loss where the objective is to optimize the euclidean distance. However, after learning the mappings, to obtain similar words the models rely on other similarity measures. This difference in measures may sometimes result in erroneous generalizations.

Also, usually to control the model properties, ℓ_1 and ℓ_2 regularizations are used. However, as the number of training observations are limited and also since ℓ_1 and ℓ_2 regularizations do not necessar-

ily generalize to unseen observations, the models might miss out on generalizing to unseen examples and rare cases.

Bilingual embeddings have also been recently utilized for the task of machine translation for better resolution of vocabulary related problems. However, a model that is not very well generalized might result in erroneous results in the context of machine translation.

In this chapter, we propose a novel technique that learns a bilinear similarity measure by capturing the compatibility between two different vector spaces of the two languages. This allows us to model the interactions in a probabilistic setting. We also apply the learned probabilistic model in the context of machine translation and conduct extensive experiments and observe a boost in the resolution of out of vocabulary words and achieve a significant improvement of 3.9 points on the BLEU scale.

Specifically, our contributions in this chapter are:

- We provide a novel model for learning bilingual projections that in turn, learns a bilinear similarity measure by exploiting the conjunctions and correlations in the vectors spaces effectively.
- We propose using low-rank regularization framework for our log-bilinear softmax based model which results in compression of bilingual embeddings.
- We apply the embeddings to the task of resolving out of vocabulary words in the task of machine translation and achieve significant improvements in the BLEU score.

5.2 BACKGROUND

Current research in inducing bilingual embeddings requires a parallel corpus [Kočiskỳ et al., 2014; Hermann and Blunsom, 2014; AP et al., 2014; Shi et al., 2015; Gouws et al., 2015] or bilingual partially sentence aligned data [Vulic and Moens, 2015; Klementiev et al., 2012; Zou et al., 2013; Wu et al., 2014] or a word to word lexicons [Mikolov et al., 2013c; Faruqui et al., 2015a; Xiao and Guo, 2014; Vulić and Moens, 2016; Dinu et al., 2015; Lazaridou et al., 2015]. Our focus is the latter models that use word lexicons.

To that extent, all the approaches are constrained by the amount of supervision that is going to be available and have to make generalizations to the vast vocabulary in both languages.

In general, we are interested in a setting where we obtain monolingual embeddings on large corpora and then use lexicons to incorporate cross lingual information, such that, given a word in a language, we are able to generate similar words in a different language.

5.3 MAPPING CONTINUOUS WORD REPRESENTATIONS USING A BILINEAR MODEL

5.3.1 *Definitions*

Let $\mathcal{E}$ and $\mathcal{F}$ be the vocabularies of the two languages, the source and the target, and let $e \in \mathcal{E}$ and $f \in \mathcal{F}$ be the words in the languages respectively. We are given with a relatively small set of source word to target word $e \rightarrow f$ dictionary. We also assume that we have access to some kind of distributed word embeddings in both languages. Let $\phi(.) \rightarrow \mathbb{R}^n$ denote the n-dimensional distributed representation of the words. We set the task such that we want to learn a model for the conditional probability distribution $\Pr(f|e)$. That is, given a word in a source language, say English (e), we want to get a conditional probability distribution of all the words in a foreign language (f).

5.3.2 *Bilinear Models*

A common approach to obtaining bilingual embeddings using a bilingual lexicon is as a linear regression based model [Mikolov et al., 2013c; Dinu et al., 2015; Vulić and Moens, 2016] in which case, a linear mapping function is learned. After learning the mapping function, the word vectors from $\mathcal{E}$ are projected on the mapping function. To estimate $\Pr(f|e)$, an external similarity measure is used to obtain the top-n similar words $\in \mathcal{F}$ given a word e. Usually, a similarity measure like cosine similarity or euclidean distance measure is preferred.

We use our conclusions from previous chapters and propose to use a bilinear formulation that learns a similarity metric by taking into consideration the correlation between the two vector spaces. We propose using a bilinear model as follows:

$$\phi(e)^\top W \phi(f) \tag{5.1}$$

here, W is the bilinear function, that is learned given a small supervision. In Equation 5.1 we are essentially getting a scalar similarity score for the pair e and f. If W is an identity matrix then Equation 5.1 reduces to a standard dot product between the two vectors.

5.3.3 *Log-Bilinear Softmax Model*

We model the task as a bilinear prediction task as proposed by Madhyastha et al. [2014] and extend it for the bilingual setting. The proposed model makes use of word embeddings on both languages with no additional features. The basic function is formulated as log-bilinear softmax model and takes the following form:

$$\Pr(f|e; W) = \frac{\exp\{\phi_s(e)^\top W \phi_t(f)\}}{\sum_{f' \in \mathcal{F}} \exp\{\phi_s(e)^\top W \phi_t(f')\}} \tag{5.2}$$

Essentially, our problem reduces to:

(a) Getting corresponding word embeddings of the vocabularies in both the languages trained on a significantly large monolingual corpus respectively, and

(b) Estimating W given a relatively small lexicon. That is, to learn W we use the source word to target word dictionary as training supervision.

Again, this is similar to our previous approach, i.e., the objective function in Equation 3.6, where we used it in the context of obtaining compatibility scores between words under a relation.

We learn W by minimizing the negative log-likelihood of the dictionary using a rank regularized objective as:

$$L(W) = -\sum_{s,t} \log(\Pr(t|s;W)) + \lambda \|W\|_*$$

where, λ is the constant that controls the capacity of W.

5.3.4 *Low-Rank Regularized Objective*

Again, to recollect from Section 3.2.5, let us consider the singular value decomposition of $W = U\Sigma V^\top$, where U and $V^\top$ are orthogonal singular vectors and Σ is an array of singular values, then we can consider $[\phi(e)^\top U]$ to be the bilingual projection of $\phi(e)$ and $[V^\top \phi(f)]$ to be the bilingual projection of $\phi(f)$. If W has rank k, where $k < n$ then $U \in \mathbb{R}^{n \times k}$ and $V^\top \in \mathbb{R}^{k \times n}$. That is if we penalize W such that it is has a low rank, we obtain a low-dimensional compressed embeddings for languages on both sides. We make use of nuclear norm regularization which is a convex relaxation for the low-rank regularized objective [Bach, 2008; Madhyastha et al., 2014, 2015].

Note, that we have no constraints on the dimensionality for $\phi(e)$ and $\phi(f)$. For illustration, we have used similar dimensionality vectors, however, the model is capable of using arbitrary dimensional vectors on both sides. In general, we empirically found that vectors initialized with similar characteristics give better performance.

5.3.5 *Motivation*

Data-driven machine translation systems are able to translate words that have been seen in the training corpora, however translating unseen words is still a major challenge for even the best performing systems.

In general, the amount of parallel data is finite (and sometimes scarce) which results in word types like named entities, domain specific content words, or infrequent terms to be absent in the training

parallel corpora. This lack of lexical information can potentially result in incomplete or erroneous translations.

There are several strands of related research that try to alleviate the effect of unseen words in translation. Previous research suggests that a significantly large number of named entities can be handled by using simple pre or post-processing techniques, like transliteration methods [Hermjakob et al., 2008; Al-Onaizan and Knight, 2002], etc.. However, a change in domain results in a significant increase in the number of unseen words. These unseen words might include a significant proportion of regular domain-specific content words.

Our focus here is to resolve unseen content words by using our proposed model. To this extent, our work is similar to Ishiwatari et al. [2016] where the authors map distributional representations using a linear regression method similar to Mikolov et al. [2013c] and insert a new feature based on cosine similarity metric into the MT system. In our work, we use a principled method to obtain a probabilistic conditional distribution of words directly and these probabilities allow us to expand the translation model for the new words.

There are other related works [Rapp, 1999; Daumé III and Jagarlamudi, 2011; Durrani and Koehn, 2014] that have explored approaches based on extracting lexicons using corpus based methods to resolve out of training vocabulary problems which are slightly constrained on the data settings.

5.3.6 *Vocabulary Expansion*

A log-linear SMT model [Och and Ney, 2002], obtains a translation f_{best} given a source sentence e and is modelled as:

$$\log(\Pr(f|e)) = \Sigma_i \lambda_i \log(g_i(e, f))$$

where, λ_i are the weights and g_i are the feature functions. This allows for the combination of several feature components.

Among the feature components, we insert the probabilistic scores for unknown words' translation options. This is weighted along with other standard log-linear model components. We tune this over a development set to obtain optimal weights.

5.4 EXPERIMENTS

We do the evaluation in two parts:

(a) We measure the quality of the obtained bilingual embeddings, and

(b) We measure the performance of the log-bilinear model in a standard machine translation setup.

5.4.1 *Embedding Quality Experiments*

We train bilingual embeddings for two language pairs: English-Spanish (En-Es) and English-French (En-Fr). For both the pairs of languages, we use the 2015 dump of wikipedia in the respected languages and the Europarl v7 corpus [Koehn et al., 2007a]. We first obtain 300-dimensional monolingual embeddings for each language. For dictionary, we use Apertium's publicly available dictionary.

EXTERNAL MODEL We use Cross-lingual Correlation based embeddings [1], proposed by Faruqui and Dyer [2014] as it uses a setting which is closest to our setting for inducing bilingual embeddings. Faruqui and Dyer [2014] project monolingual embeddings using canonical correlation analysis [Hotelling, 1936], such that the projected vectors are as close as possible for word pairs in the lexicon. We use the same parametric settings as is followed in Upadhyay et al. [2016]. However, after projection, the bilingual vectors are reduced from an initial 300-dimensional vector to a 100-dimensional vector.

We will first evaluate the quality of word embeddings using word similarity metric and test the performance of bilingual word embeddings in the context of a syntactic task.

WORD SIMILARITY FOR ENGLISH This task basically evaluates the word similarity in the embeddings space correlates with the human evaluations. This is measured using Spearman's rank correlation coefficient. Following Upadhyay et al. [2016] we use SimLex dataset [Hill and Korhonen, 2014] that captures word similarity notion exclusively.

We compress the embeddings by projecting the monolingual embeddings on to the top 50, top 100, top 200 and 300 singular vectors respectively. Table 5.1 shows the performance of the bilingual embeddings both on our bilinear model as well as using the cross-lingual CCA based model. We notice that the projected embeddings in some cases have better correlations than the monolingual embeddings, however, they are not much superior to the standard monolingual embeddings.

We also notice that our compressed English embeddings in both language pairs achieve almost comparative scores.

LangPair	MonoLing	50-Bil	100-Bil	200-Bil	300-Bil	CL-CCA
En-Fr	0.39	0.36	0.38	0.39	0.40	0.39
En-Es	0.37	0.34	0.37	0.39	0.38	0.37

Table 5.1: Word Similarity Score

1 We obtained the source code here: https://github.com/mfaruqui/crosslingual-cca

CROSS LINGUAL SYNTACTIC DEPENDENCY PARSING We want to test the performance of the bilingual embeddings on the task of dependency parsing, where we are interested in seeing if the bilingual projected vectors perform better than the monolingual vectors. Again, we follow Upadhyay et al. [2016] and perform the experiments in the similar settings and we use the system by Gouws et al. [2015] [2]. We use the universal dependency treebank version 2.0 for the evaluation.

We notice a similar trend as before. The 50-dimensional compression is underperforming, while the others are competitive.

Lang Pair	MonoLing	50 bilinear	100 bilinear	300 Bilinear	CL-CCA*
En-**Fr**	78.77	78.64	79.54	**80.63**	80.11
En-**Es**	80.12	79.37	80.08	81.17	**81.57**

Table 5.2: Labelled Attachment Score trained on English and tested on language French and Spanish

5.4.2 *Experiments with the model on SMT*

We now focus on the application of our proposed log bilinear model to resolve out of vocabulary words for machine translation.

DATA AND SYSTEM SETTINGS. For estimating the word embeddings we use the CBOW algorithm as implemented in the `Word2Vec` package [Mikolov et al., 2013b][3] using a 5-token window. We obtain 300 dimension vectors for English and Spanish from a Wikipedia dump of 2015[4], and the Quest data[5] which includes subcorpora such as United Nations and Europarl. The final corpus contains 2.27 billion tokens for English and 840 million tokens for Spanish. We obtain a coverage of 97% of the words in our test sets. We also remove any occurrence of sentences from the test set that are contained in our corpus, and avoid any transduction based knowledge transfer.

To train the log-bilinear softmax based model, we use the dictionary from the `Apertium project`[6] [Forcada et al., 2011]. The dictionary contains 37651 words, we used 70% of them for training the log-bilinear model and 30% as a development set for model selection. The average precision @1 was 85.66% for the best model over the dev set.

We build a state-of-the-art phrase-based SMT system trained on the standard Europarl corpus for the English-to-Spanish language pair. We use a 5-gram language model that is estimated on the target

2 we obtain the system here: https://github.com/jiangfeng1124/acl15-clnndep
3 https://code.google.com/archive/p/word2vec/
4 Dumps downloaded in January 2015 from https://dumps.wikimedia.org.
5 http://goo.gl/72LLXN
6 The bilingual dictionary can be downloaded here: http://goo.gl/TjH31q.

Table 5.3: OOVs on the dev and test sets.

	Sent.	Tokens	OOV_{all}	OOV_{CW}
NewsDev	3003	72988	1920 (2.6%)	378 (0.5%)
NewsTest	3000	64810	1590 (2.5%)	296 (0.5%)
WikiTest	500	11069	798 (7.2%)	201 (1.8%)

side of the corpus using interpolated Kneser-Ney discounting with `SRILM` [Stolcke, 2002]. Additional monolingual data available within Quest corpora are used to build a larger language model with the same characteristics. Word alignment is done with `GIZA++` [Och and Ney, 2003] and both phrase extraction and decoding are done with the `Moses` package [Koehn et al., 2007b].

At decoding time, `Moses` allows to include additional translation pairs with their associated probabilities to selected words via `XML` mark-up. We take advantage of this feature to add our probabilistic estimations to each OOV. Since, by definition, OOV words do not appear in the parallel training corpus, they are not present in the translation model either and the new translation options only interact with the language model.

The optimization of the weights of the model with the additional translation options is trained with MERT [Och, 2003] against the BLEU [Papineni et al., 2002] evaluation metric on the NewsCommentaries 2012[7] (NewsDev) set. We test our systems on the NewsCommentaries 2013 set (NewsTest) for an in-domain evaluation and on a test set extracted from Wikipedia by Smith et al. [2010] for an out-of-domain evaluation (WikiTest).

The *domainess* of the test set is established with respect to the number of OOVs. Table 5.3 shows the exact numbers of these sets, paying special attention to the OOVs in the basic SMT system. Less than a 3% of the tokens are OOVs for News data (OOV_{all}), whereas it is more than a 7% for Wikipedia's. In our experiments, we distinguish between OOVs that are named entities and the rest of content words (OOV_{CW}). Only about 0.5% (NewsTest) and 1.8% (WikiTest) of the tokens fall into this category, but we show that they are relevant for the final performance.

EVALUATION. We consider two baseline systems, the first one does not output any translation for OOVs (*noOOV*), it just ignores the token; the second one outputs a verbatim copy of the unseen word as a translation (*verbatimOOV*). Table 5.4 shows the performance of these systems under three widely used evaluation metrics TER [Snover et al., 2006], BLEU and METEOR-paraphrase (MTR) [Banerjee and Lavie, 2005]. Including the verbatim copy improves all the lexical

7 http://www.statmt.org/wmt13/translation-task.html

	NewsTest			WikiTest		
	TER	BLEU	MTR	TER	BLEU	MTR
noOOV	58.21	21.94	45.79	61.26	16.24	38.76
verbatimOOV	57.90	22.89	47.06	58.55	21.90	45.77
BWE	58.33	22.23	45.76	58.38	21.96	44.84
BWE_{CW50}	57.66	23.09	47.14	56.19	24.16	48.49
BWE_{CW10}	57.85	23.06	47.11	55.64	24.71	49.05
BLM	55.37	25.83	**49.19**	52.60	30.63	51.04
BLM+BWE	55.89	24.92	47.84	51.02	32.20	52.09
$BLM+BWE_{50}$	55.55	25.61	49.01	49.50	33.94	54.93
$BLM+BWE_{10}$	**55.31**	**25.86**	49.04	**49.12**	**34.58**	**55.52**

Table 5.4: Automatic evaluation of the translation systems. The best system is bold-faced.

evaluation metrics. Especially, we observe that for named entities and acronyms (the 80% of OOVs in our sets), this is a hard baseline to beat. This is because, in most cases the same word is the correct translation (e.g. Messi, PHP, Sputnik,...).

We now enrich the systems with information gathered from the large monolingual corpora in two ways, using a bigger language model (*BLM*), i.e., a language model trained on a very large corpora, and using our newly proposed log-bilinear model that uses word embeddings (*BWE*). BLMs are very important to improve the fluency of the translations, however, they may not be helpful for resolving out-of-vocabulary words. On the other hand, BWEs are important to make available to the decoder new vocabulary on the topic of the otherwise OOVs. Given the large percentage of named entities in the test sets (Table 5.3), our models add the source word as an additional option to the list of target words to mimic the *verbatimOOV* system.

Table 5.4 includes seven systems with the additional monolingual information. Three of them add, at decoding time, the top-n translation options given by the BWE for an OOV. *BWE* system uses the top-50 for all the OOVs, BWE_{CW50} also uses the top-50 but only for content words other than named entities[8], and BWE_{CW10} limits the list to 10 elements. *BLM* is the same as the baseline system *verbatimOOV* but with the large language model. *BLM+BWE*, $BLM+BWE_{50}$ and $BLM+BWE_{10}$ combine the three BWE systems with the large language model.

A large number of unseen words in the NewsTest are mostly named entities, using BWEs to translate all the words, including named enti-

8 We consider a named entity any word that begins with a capital letter and is not after a punctuation mark, and any fully capitalized word.

Table 5.5: Top-n list of translations obtained with the bilingual embeddings.

GALAXY	NYMPHS	STUART	FOLKSONG
galaxia	**ninfas**	William	música
planeta	ninfa	Henry	**folclore**
universo	crías	John	literatura
planetas	diosa	Charles	himno
galaxias	dioses	Thomas	folklore
...	...	Estuardo (#48)	**canción (#7)**

ties, barely improves the translation. Also, the richness in vocabulary, consisting of many names, adds noise to the decoder. We observe that the improvements are moderate in the NewsTest (in-domain dataset), this is mostly because, the differences in the probability of the BWE translation options are very small owing to the candidates being named entities. We also see that this affects the overall integration of the scores into the decoder and also induces ambiguity in the system. On the other hand, we observe that the decoder benefits from the information on content words, especially for the out-of-domain WikiTest set, given the constrained list of alternative translations ($BWE_{\text{CW}_{10}}$ achieves 2.75 BLEU points of improvement).

The addition of the large language model improves the results significantly. When combined with the BWEs we observe that the BWEs clearly help in the translation of WikiTest but do not seem as relevant in the in-domain set. We also achieve a statistically significant improvement of 3.9 points of BLEU with the BLM and BWE combo system in WikiTest ($p < 0.001$). The number of translation options in the list is also relevant, we see that for $BWE_{\text{CW}_{50}}$ we have an improvement of 3.3 points on BLEU. We also observe that the results are consistent among different metrics.

We have further manually evaluated the translation of WikiTest using $BWE_{\text{CW}_{50}}$. We obtained an accuracy of a 68%, that is, the BWE gives the correct translation option at least 68% of the times. The other 32% of the time, it fails as the words in the translated language happened to be either multiwords or named entities. In table 5.5 we observe some of the these examples. In the first two examples, *galaxy* and *nymphs* are nouns, where we obtain the first option as the correct translation. The problem is harder for named entities. We observe in the table, the name *Stuart* in English has *William* as most probable translation in Spanish, the correct translation *Estuardo*, however appears as the 48th choice. Our model is also unable to generate multiword expressions, as shown in the table for the English word *folksong*,

the correct translation being *canción folk*. This would need two words in Spanish in order to be translated correctly, however, our model does obtain words: *canción* and *folclore* as the most probable translation options.

5.5 CONCLUSIONS

We have presented a method for resolving unseen words in SMT that performs vocabulary expansion by using a simple log-bilinear softmax based model. The addition of translation options to a mere 1.8% of the words has allowed the system to obtain a relative improvement of a 13% in BLEU (3.9 points) for out-of-domain data. For in-domain data, where the number of content words is small, improvements are moderate. We would like to further study the repercussion of this simple method on diverse and more distant language pairs and how the form of the bilinear loss function affects the quality of the bilingual word embeddings.

6

MAPPING UNSEEN WORDS TO TASK-TRAINED EMBEDDINGS SPACE

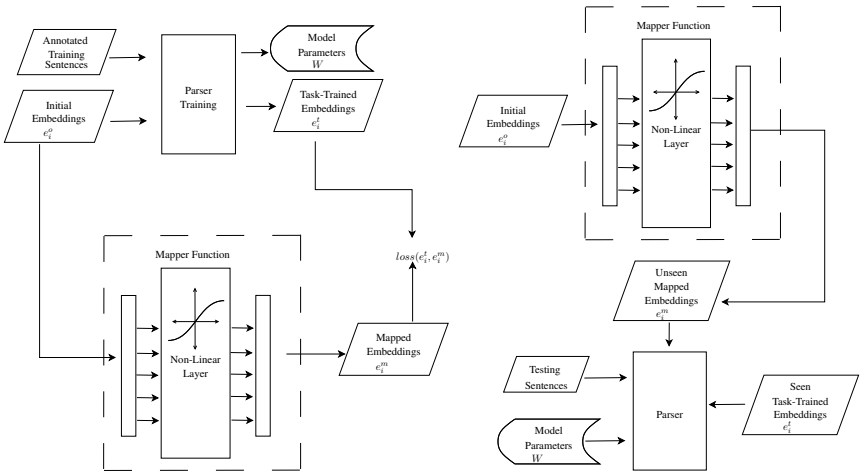
In this chapter, we describe our neural network-based mapping function that takes initial word embeddings (learned using unsupervised techniques over a large unannotated dataset) and maps them to task-specific embeddings that are trained for the given task, via a multi-loss objective function. Moreover, due to the efficiency of training our mapper, we can tune its hyperparameters to optimize performance on each domain of interest, thereby achieving some of the benefits of domain adaptation.

We start by describing our model for mapping unseen representations. In the subsequent section, we survey some related approaches. We then show the effectiveness of our mapping approach on the task of dependency parsing on a large set of diverse domains, e.g., news, the Web, and speech. We also show the downstream effects of our mapper on the task of sentiment classification, using the recently proposed dependency parse-based tree long short-term memory (LSTM) network of Tai et al. [2015]. In all domains and tasks, our method significantly reduces the number of unseen words (i.e., words without task-trained embeddings) and achieves significant performance improvements over the unmapped baselines. Most of this work is published in Madhyastha et al. [2016].

6.1 MAPPING UNSEEN REPRESENTATIONS

In many NLP tasks, state-of-the-art systems achieve very good performance, but only when restricted to standard and heavily edited datasets [Petrov et al., 2010]. For example, while state-of-the-art accuracies exceed 97 percent for part-of-speech tagging and 90 percent for dependency parsing, performance on non-standard, real-world datasets is substantially worse, dropping by nearly 10 percent absolute [Foster, 2010; Petrov and McDonald, 2012b]. A major cause of this drop is words that do not appear in the annotated training data but appear in unseen test data, whether in the same domain or in a new domain. We refer to such out-of-training-vocabulary (OOTV) words as *unseen* words. Supervised NLP systems often make errors on unseen words and, in structured tasks like dependency parsing, this can lead to other cascading errors in the same sentence, producing nonsensical output structures.

As we have seen in Chapter 2, continuous vector word representations, or embeddings, have shown promise in a variety of NLP



(a) Mapper Training

(b) test-time: parsing with mapped embeddings

Figure 6.1: system pipeline

task [Turian et al., 2010a; Collobert et al., 2011b; Anderson et al., 2013; Bansal et al., 2014a]. The typical assumption of word embedding learning is that words with similar statistical properties in a large dataset are similar in meaning. Using embeddings as features in NLP systems can help counter the effects of data sparsity [Necsulescu et al., 2015]. However, the quality of such embeddings has been found to be heavily task-dependent [Bansal et al., 2014b].

There is a great deal of work on updating embeddings during supervised training to make them more task-specific, whether through back-propagation or other techniques [Kalchbrenner et al., 2014; Qu et al., 2015b; Chen and Manning, 2014b]. These task-trained embeddings have shown encouraging results but raise some concerns, especially: (1) for infrequent words, the updated representations may be prone to overfitting, and (2) many words in the test data are not contained in the training data at all. In the latter case, at test time, most trained systems either fall back to some generic, single representation for all unknown words or use the initial representation (typically derived from unlabeled data) [Søgaard and Johannsen, 2012; Collobert et al., 2011b]. Neither of these choices is satisfactory: A single unknown word vector may lump too many words together, while the initial representations may be in a space which is not comparable to the trained embedding space.

In this section, we present our approach, including an overview of the pipeline, the mapper architecture, and the training/tuning components (e.g., loss function, regularization, and thresholds).

6.1.1 *Pipeline Overview and Definitions*

Let $\mathcal{V} = \{w_1, \ldots, w_V\}$ be the vocabulary of word types in a large, unannotated corpus. Let e_i^o denote the initial (original) embedding of word w_i computed from this corpus. The initial embeddings are typically learned in an unsupervised way, but for our purposes, they can be any initial embeddings. We define **unseen** words as those in the set $\mathcal{V} \setminus \mathcal{T}$. While our approach is general, for concreteness, we consider the task of dependency parsing, so the annotated data consists of sentences paired with dependency trees. We assume a dependency parser that learns task-specific word embeddings e_i^t for word $w_i \in \mathcal{T}$, starting from the original embedding e_i^o. In this work, we use the Stanford neural dependency parser [Chen and Manning, 2014b].

The goal of the mapper is as follows. We are given a training set of N pairs of initial and task-trained embeddings $\mathcal{D} = \left\{ \left(e_1^o, e_1^t\right), \ldots, \left(e_N^o, e_N^t\right) \right\}$, and we want to learn a function G that maps each initial embedding e_i^o to be as close as possible to its corresponding output embedding e_i^t. We denote the mapped embedding e_i^m, i.e., $e_i^m = G\left(e_i^o\right)$.

Figure 6.1a describes the training procedure of the mapper. We use a supervised parser which is trained on an annotated dataset and initialized with pre-trained word embeddings e_i^o. The parser uses back-propagation to update these embeddings during training, producing task-trained embeddings e_i^t for all $w_i \in \mathcal{T}$. After we train the parser, the mapping function G is trained to map an initial word embedding e_i^o to its parser-trained embedding e_i^t. At test (or development) time, we use the trained mapper G to transform the original embeddings of unseen test words to the parser-trained space (see Figure 6.1b). When parsing held-out data, we use the same parser model parameters (W) as shown in Figure 6.1b. The only difference is that now some of the word embeddings (i.e., for unseen words) have changed to mapped ones.

6.1.2 *Mapper Architecture*

Our proposed mapper is a multi-layer that takes an initial word embedding as input and outputs a mapped representation of the same dimensionality. In particular, we use a single hidden layer

$$G(e_i^o) = W_2(\text{hard tanh}(W_1 e_i^o + b_1)) + b_2 \tag{6.1}$$

where W_1 and W_2 are parameter matrices and b_1 and b_2 are bias vectors.

The 'hardtanh' non-linearity is the standard 'hard' version of hyperbolic tangent:

$$\text{hard}\tanh(z) = \begin{cases} -1 & \text{if } z < -1 \\ 1 & \text{if } z > 1 \end{cases}$$

In preliminary experiments, we compared with other non-linear functions (sigmoid, tanh, and ReLU), as well as with zero and more than one non-linear layer. We found that hard tanh is computationally cheaper and performed better than the other non-linearities, and that the fewer or more non-linear layers did not improve performance.

6.1.3 *Loss Function*

We use a weighted, multi-loss regression approach, optimizing a weighted sum of mean squared error and mean absolute error:

$$loss(y, \hat{y}) = \\ \alpha \sum_{j=1}^{n} |y_j - \hat{y}_j| + (1-\alpha) \sum_{j=1}^{n} |y_j - \hat{y}_j|^2 \tag{6.2}$$

where $y = e_i^t$ (the ground truth) and $\hat{y} = e_i^m$ (the prediction) are n-dimensional vectors. This multi-loss approach seeks to make both the conditional *mean* of the predicted representation close to the task-trained representation (via the squared loss) and the conditional *median* of the predicted representation close to the task-trained one (via the mean absolute loss). A weighted multi-criterion objective allows us to avoid making strong assumptions about the optimal transformation to be learned. We tune the hyperparameter α on domain-specific held-out data.

For optimization, we use batch limited memory BFGS (L-BFGS) [Liu and Nocedal, 1989]. In preliminary experiments comparing with stochastic optimization, we found L-BFGS to be more stable to train and easier to check for convergence (as has recently been found in other settings as well [Ngiam et al., 2011]).

6.1.4 *Regularization*

We use elastic net regularization [Liu and Nocedal, 1989], which linearly combines ℓ_1 and ℓ_2 penalties on the parameters to control the capacity of the mapper function. This equates to minimizing:

$$F(\theta) = L(\theta) + \lambda_1 \|\theta\|_1 + \frac{\lambda_2}{2} \|\theta\|_2^2$$

where θ is the full set of mapper parameters and $L(\theta)$ is the loss function (Eq. 6.2 summed over mapper training examples). We tune the

hyperparameters of the regularizer and the loss function separately for each task, using a task-specific development set. This gives us additional flexibility to map the embeddings for the domain of interest, especially (e.g., newswire) and we want to use the parser on a new domain (e.g., email). We also tried dropout-based regularization [Srivastava et al., 2014] for the non-linear layer but did not see any significant improvement.

6.1.5 *Mapper-Parser Thresholds*

Certain words in the parser training data $\mathcal{T}$ are very infrequent, which may lead to inferior task-specific embeddings e_i^t learned by the parser. We want our mapper function to be learned on high-quality task-trained embeddings. After learning a strong mapping function, we can use it to remap the inferior task-trained embeddings.

We thus consider several frequency thresholds that control which word embeddings to use to train the mapper and which to map at test time. Below are the specific thresholds that we consider:

MAPPER-TRAINING THRESHOLD (τ_t) The mapper is trained only on embedding pairs for words seen at least τ_t times in the training data $\mathcal{T}$.

MAPPING THRESHOLD (τ_m) For test-time inference, the mapper will map any word whose count in $\mathcal{T}$ is less than τ_m. That is, we discard parser-trained embeddings e_i^t of these infrequent words and use our mapper to map the initial embeddings e_i^o instead.

PARSER THRESHOLD (τ_p) While training the parser, for words that appear fewer than τ_p times in $\mathcal{T}$, the parser replaces them with the 'unknown' embedding. Thus, no parser-trained embeddings will be learned for these words.

In our experiments, we explore a small set of values from this large space of possible threshold combinations (detailed below). We consider only relatively small values for the mapper-training (τ_t) and parser thresholds (τ_p) because as we increase them, the number of training examples for the mapper decreases, making it harder to learn an accurate mapping function.

6.2 RELEVANT RELATED WORK

The most common approach to resolving unseen words is to replace them with a special unknown word token [Søgaard and Johannsen, 2012; Chen and Manning, 2014b; Collobert et al., 2011b]. The representation for the unknown token is either learned specifically or computed from a selection of rare words, for example by averaging

their embedding vectors. This approach could be problematic since all unseen words are mapped to the same vector, irrespective of their syntactic or semantic categories.

There are several threads of prior work using character-level information for unseen word forms. Some combine unsupervised morphological analysis with compositional neural network architectures [Luong et al., 2013b; Botha and Blunsom, 2014]. Ling et al. [2015] and Ballesteros et al. [2015] use long short-term memory recurrent neural networks to embed character sequences. Others use convolutional neural networks on character streams [Labeau et al., 2015; Kim et al., 2016; Zhang and LeCun, 2015]. Huang and Harper (2009; 2011) use a heuristic based on emission probabilities of individual characters in unknown words.

There is also a great deal of work using morphological information for rare or unseen words [Candito and Crabbé, 2009; Habash, 2009; Marton et al., 2010; Seddah et al., 2010; Attia et al., 2010]. Other work has focused on using contextual information, such as using n-gram based sequence models or web data [Bansal and Klein, 2011]. [Keller and Lapata, 2003] use the web to obtain frequency information for unseen words.

Dyer et al. [2015b] represent each word using an embedding learned during parser training concatenated with a fixed embedding from a neural language model, which provides a larger vocabulary. While they still use an unknown word token for singletons, the pretrained embeddings from the neural language model provide additional information about the unknown word. Other work has also found improvements by combining pretrained, fixed embeddings with task-trained embeddings [Kim, 2014; Paulus et al., 2014]. Also relevant are approaches developed specifically to handle large target vocabularies (including many rare words) in neural machine translation systems [Jean et al., 2015; Luong et al., 2015; Chitnis and DeNero, 2015].

Closely related to our approach is that of Tafforeau et al. [2015], which also tries to map an initial, unsupervised word embedding space to the word embedding space learned during supervised training. Their method generates updated embeddings for unseen words by combining the embeddings of their k nearest neighbors. In Section 6.4, we show that our approach outperforms this k-NN approach. Another related technique proposed by Kiros et al. [2015] learns a linear mapping from an initial embedding space to their encoder's vocabulary space by solving an unregularized linear regression problem. Our approach differs in that it can learn a non-linear mapping, and we also learn separate mappings for each domain of interest, tuning the mapper for each domain. We also empirically evaluate the effect of performing this mapping, showing statistically significant improvements. To our knowledge, these are the only approaches that

handle unseen words by explicitly mapping initial representations to a task-trained space.

Our work is also somewhat related to domain adaptation for dependency parsing, which has been extensively studied in recent years [McDonald and Nivre, 2007; Nilsson et al., 2007]. The goal of this task is to adapt an existing parser to a target domain with little or no annotated data. Previous work has used co-training [Cohen et al., 2012], word distribution features [Koo et al., 2008; Bansal et al., 2014b; Weiss et al., 2015], and self-training [McClosky et al., 2006].

Our simple approach learns to directly map initial word-level embeddings to the task-trained embedding space. No character-level compositional model or morphological information is needed. Since the initial embeddings are obtained from a large corpus in an unsupervised manner, data sparseness issues can be mitigated by enlarging this corpus. Tuning the hyperparameters of our mapper on small domain-specific development sets helps us to adapt the transformation to the target domain.

6.3 EXPERIMENTAL SETUP

In this section, we describe our primary parsing setup, embeddings, and datasets. We also describe the setup for a downstream task: sentiment analysis using a neural network model based on dependency trees. Finally, we discuss the settings for the mapper.

6.3.1 *Dependency Parser*

We use the feed-forward neural network dependency parser of Chen and Manning [2014b]. In all our experiments (unless stated otherwise), we use the default arc-standard parsing configuration and hyperparameter settings. For evaluation, we compute the percentage of words that get the correct head, reporting both unlabeled attachment score (UAS) and labeled attachment score (LAS). LAS additionally requires the predicted dependency label to be correct. To measure statistical significance, we use a bootstrap test [Efron and Tibshirani, 1986] with 100K samples.

6.3.2 *Pre-Trained Word Embeddings*

We use the 100-dimensional GloVe word embeddings from Pennington et al. [2014b]. These were trained on Wikipedia 2014 and the Gigaword v5 corpus and have a vocabulary size of approximately 400,000.We have also experimented with the downloadable 50-dimensional SENNA embeddings from [Collobert et al., 2011b] and with `word2vec` [Mikolov et al., 2013a] embeddings that we trained ourselves; in preliminary ex-

periments the GloVe embeddings performed best, so we use them for all experiments below.

6.3.3 *Datasets*

We consider a number of datasets with varying rates of OOTV words. We define the OOTV rate (or, equivalently, the unseen rate) of a dataset as the percentage of the vocabulary (types) of words occurring in the set that was not seen in training.

WALL STREET JOURNAL (WSJ) AND ONTONOTES-WSJ We conduct experiments on the Wall Street Journal portion of the English Penn Treebank dataset [Marcus et al., 1993a]. We follow the standard splits: sections 2-21 for training, section 22 for validation, and section 23 for testing. We convert the original phrase structure trees into dependency trees using Stanford Basic Dependencies [De Marneffe and Manning, 2008] in the Stanford Dependency Parser. The POS tags are obtained using the Stanford POS tagger [Toutanova et al., 2003] in a 10-fold jackknifing setup on the training data (achieving an accuracy of 96.96%). The OOTV rate in the development and test sets is approximately 2-3%.

We also conduct experiments on the OntoNotes 4.0 dataset (which we denote OntoNotes-WSJ). This dataset contains the same sentences as the WSJ corpus (and we use the same data splits) but has significantly different annotations. The OntoNotes-WSJ training data is used for the Web Treebank test experiments. We perform the same pre-processing steps as for the WSJ dataset.

WEB TREEBANK We expect our mapper to be most effective when parsing held-out data with many unseen words. This often happens when the held-out data is drawn from a different distribution than the training data. For example, when training a parser on the newswire and testing on web data, many errors occur due to differing patterns of syntactic usage and unseen words [Foster et al., 2011; Petrov and McDonald, 2012b; Kong et al., 2014; Wang et al., 2014].

We explore this setting by training our parser on OntoNotes-WSJ and testing on the Web Treebank [Petrov and McDonald, 2012b], which includes five domains: answers, email, newsgroups, reviews, and weblogs. Each domain contains approximately 2000-4000 manually annotated syntactic parse trees in the OntoNotes 4.0 style. As before, we convert the phrase structure trees to dependency trees using Stanford Basic Dependencies. The parser and the mapper hyperparameters were tuned separately on the development set for each domain. The test and development sets for each domain contain approximately 1000-2500 sentences. We use our mapper to map unseen words in the development and test sets in each domain. The unseen rate is typi-

cally 6-10% in the domains of the Web Treebank. We train a separate mapper for each domain, tuning mapper hyperparameters separately for each domain using the development sets. In this way, we obtain some of the benefits of domain adaptation for each target domain.

SWITCHBOARD SPEECH CORPUS The NXT Switchboard speech corpus [Calhoun et al., 2010] contains annotated parses of spoken telephone conversations. We obtain ground truth dependencies from phrase structure trees using the Stanford converter as above, as also done by Honnibal and Johnson [2014]. We perform their other preprocessing steps of lowercasing the text, removing punctuation, and removing partial utterances and one-token sentences. Since the current version of the Stanford parser cannot perform non-monotonic parsing,[1] we also remove disfluent utterances in such a way that we get a purely non-disfluent speech dataset. We use the standard train/development/test splits of Charniak and Johnson [2001].

DOWNSTREAM TASK: SENTIMENT ANALYSIS WITH DEPENDENCY TREE LSTMS We also perform experiments to analyze the effects of embedding mapping on a downstream task, in this case, sentiment analysis using the Stanford Sentiment Treebank [Socher et al., 2013b]. We use the dependency tree long short-term memory network (Tree-LSTM) proposed by Tai et al. [2015], simply replacing their default dependency parser with our version that maps unseen words. The dependency parser is trained on the WSJ corpus and mapped using the WSJ development set. We use the same mapper that was optimized for the WSJ development set, without further hyperparameter tuning for the mapper. For the Tree-LSTM model, we use the same hyperparameter tuning as described in Tai et al. [2015]. We use the standard train/development/test splits of 6820/872/1821 sentences for the binary classification task and 8544/1101/2210 for the fine-grained task.

6.3.4 *Mapper Settings and Hyperparameters*

The initial embeddings given to the mapper are the same as the initial embeddings given to the parser. These are the 100-dimensional GloVe embeddings mentioned above. The output dimensionality of the mapper is also fixed to 100. All model parameters of the mappers are initialized to zero. We set the dimensionality of the non-linear layer to 400 across all experiments. The model parameters are optimized by maximizing the weighted multiple-loss objective using L-BFGS with elastic-net regularization (Section 6.1). The hyperpa-

1 The arc-standard algorithm, in the Stanford parser, is a monotonic parsing algorithm where once an action has been performed, subsequent actions must be consistent with it [Honnibal and Johnson, 2014]. This does not work well with disfluent utterances.

Lower OOTV word rate				
	WSJ	OntoNotes	Switchboard	Avg.
UAS	91.85→92.21	90.17→90.49	89.12→89.41	90.38→90.70
LAS	89.49→89.73	87.68→87.92	86.58→86.77	87.92→88.14
OOTV %	2.72→1.45	2.72→1.4	2.1→0.98	–
OOTV subset	89.88→90.51	89.27→89.81	88.22→89.03	89.12→89.78
#Sents	337	329	437	–

Higher OOTV word rate						
	Answers	Emails	Newsgroups	Reviews	Weblogs	Avg.
UAS	82.67→83.21	81.76→82.42	84.68→85.13	84.25→84.99	87.73→88.43	84.22→84.84
LAS	78.98→79.59	77.93→78.56	81.88→82.71	81.26→81.92	85.68→86.29	81.01→81.81
OOTV %	8.53→1.22	10.56→3.01	10.34→1.04	6.84→0.73	8.45→0.38	–
OOTV subset	80.88→81.75	79.29→81.02	82.54→83.71	81.17→82.22	86.43→87.31	82.06→83.20
#Sents	671	644	579	632	541	–

Table 6.1: Results of dependency parsing on various treebanks. Entries of the form A→B give results for parsing without mapped embeddings (A) and with mapped embeddings (B). "OOTV %" entries A→B indicate that A% of the test set vocabulary was unseen in the parser training, and B% remain unknown after mapping the embeddings. "OOTV subset" refers to UAS measured on the subset of the test set sentences that contain at least one OOTV word, and "#Sents" gives the number of sentences in this subset.

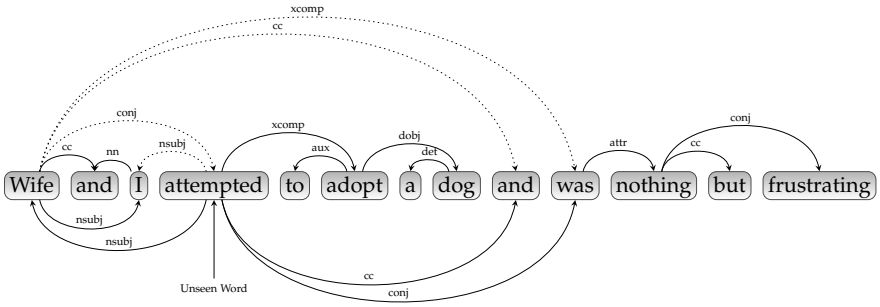
rameters include the relative weight of the two objective terms (α) and the regularization constants (λ_1, λ_2). For α, we search over values in $\{0, 0.1, 0.2, \ldots, 1\}$. For each of λ_1 and λ_2, we consider values in $\{10^{-1}, 10^{-2}, \ldots, 10^{-9}, 0\}$. The hyperparameters are tuned via grid search to maximize the UAS on the development set.

6.4 RESULTS AND ANALYSIS

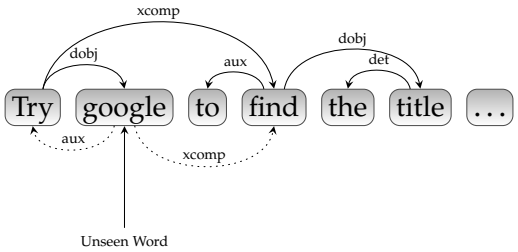
6.4.1 *Results on WSJ, OntoNotes, and Switchboard*

The upper half of Table 6.1 shows our main test results on WSJ, OntoNotes, and Switchboard, the low-OOTV rate datasets. Due to the small initial OOTV rates (<3%), we only see modest gains of 0.3-0.4% in UAS, with statistical significance at $p < 0.05$ for WSJ and OntoNotes and $p < 0.07$ for Switchboard. The initial OOTV rates are cut in half by our mapper, with the remaining unknown words largely being numerical strings and misspellings.[2] When only considering test sentences containing OOTV words (the row labeled "OOTV subset"), the gains are significantly larger (0.5-0.8% UAS at $p < 0.05$).

2 We could potentially train the initial embeddings on a larger corpus or use heuristics to convert unknown numbers and misspellings to forms contained in our initial embeddings.



(a) We obtain correct attachments and correct tree after the mapper maps the unseen word 'attempted'



(b) The mapper incorrectly maps 'google', resulting in wrong attachments and wrong tree

Figure 6.2: Examples where the mapper helps and hurts: In the above examples the top arcs are before mapping and bottom ones are after mapping; dotted lines refer to incorrect attachment.

6.4.2 *Results on Web Treebank*

The lower half of Table 6.1 shows our main test results on the Web Treebank's five domains, the high-OOTV rate datasets. As expected, the mapper has a much larger impact when parsing these out-of-domain datasets with high OOTV word rates.[3]

The OOTV rate reduction is much larger than for the WSJ-style datasets, and the parsing improvements (UAS and LAS) are statistically significant at $p < 0.05$. On subsets containing at least one OOTV word (that also has an initial embedding), we see an average gain of 1.14% UAS (see row labeled "OOTV subset"). In this case, all improvements are statistically significant at $p < 0.02$. We observe that the relative reduction in OOTV% for the Web Treebanks is larger than for the WSJ, OntoNotes, or Switchboard datasets. In particular, we are able to reduce the OOTV% by 71-95% relative. We also see the intuitive trend that larger relative reductions in OOTV rate correlate with larger accuracy improvements.

3 As stated above, we train the parser on the OntoNotes dataset, but tune mapper hyperparameters to maximize parsing performance on each development section of the Web Treebank's five domains. We then map the OOTV word vectors on each test set domain using the learned mapper for that domain.

Fine-Grained	Binary
48.4→49.5	85.7→ 86.1

Table 6.2: Improvements on Stanford Sentiment Treebank test set using our parser with the Dependency Tree-LSTM.

6.4.3 *Downstream Results*

We now report results using the Dependency Tree-LSTM of Tai et al. [2015] for sentiment analysis on the Stanford Sentiment Treebank. We consider both the binary (positive/negative) and fine-grained classification tasks ({very negative, negative, neutral, positive, and very positive}). We use the implementation provided by Tai et al. [2015], changing only the dependency parses that are fed to their model. The sentiment dataset contains approximately 25% OOTV words in the training set vocabulary, 5% in the development set vocabulary, and 9% in the test set vocabulary. We map unseen words using the mapper tuned on the WSJ development set. We use the same Dependency Tree-LSTM experimental settings as Tai et al. Results are shown in Table 6.2. We improve upon the original accuracies in both binary and fine-grained classification. We also reduce the OOTV rate from 25% in the training set vocabulary to about 6%, and from 9% in the test set vocabulary down to 4%.

6.4.4 *Effect of Thresholds*

We also experimented with different values for the thresholds described in Section 6.1. For the mapping threshold τ_m, mapper-training threshold τ_t, and parser threshold τ_p, we consider the following four settings:

$$\begin{aligned}
\mathbf{t_1} &: \tau_m = \tau_t = \tau_p = 1 \\
\mathbf{t_3} &: \tau_m = \tau_t = \tau_p = 3 \\
\mathbf{t_5} &: \tau_m = \tau_t = \tau_p = 5 \\
\mathbf{t_\infty} &: \tau_m = \infty, \tau_p = \tau_t = 5
\end{aligned}$$

Using $\tau_m = \infty$ corresponds to mapping all words at test time, even words that we have seen many times in the training data and learned task-specific embeddings for.

We report the average development set UAS over all Web Treebank domains in Table 6.3. We see that $\mathbf{t_3}$ performs best, though settings $\mathbf{t_1}$ and $\mathbf{t_5}$ also improve over the baseline. At threshold $\mathbf{t_3}$ we have approximately 20,000 examples for training the mapper, while at threshold $\mathbf{t_5}$ we have only about 10,000 examples. We see a performance drop at $\mathbf{t_\infty}$, so it appears better to directly use the task-specific embeddings for words that appear frequently in the training data. In

Baseline	t_1	t_3	t_5	t_∞
84.11	84.89	84.97	84.81	84.14

Table 6.3: Average Web Treebank development UAS at different threshold settings.

other results reported in this chapter, we used t_3 for the Web Treebank test sets and t_1 for the rest.

6.4.5 *Effect of Weighted Multi-Loss Objective*

We analyzed the results when varying α, which balances between the two components of the mapper's multi-loss objective function. We found that, for all domains except Answers, the best results are obtained with some α between 0 and 1. The optimal values outperformed the cases with $\alpha = 0$ and $\alpha = 1$ by 0.1-0.3% UAS absolute. i.e., the mapper preferred mean squared error. For other domains, the optimal α tended to be within the range $[0.3, 0.7]$.

6.4.6 *Comparison with Related Work*

We compare to the approach presented by Tafforeau et al. [2015]. They propose to refine embeddings for unseen words based on the relative shifts of their k nearest neighbors in the original embeddings space. Specifically, they define "artificial refinement" as:

$$\phi_r(t) = \phi_o(t) + \sum_{k=1}^{K} \alpha_k(\phi_r(n_k) - \phi_o(n_k)) \tag{6.3}$$

where $\phi_r(.)$ is the vector in the refined embedding space and $\phi_o(.)$ is the vector in the original embedding space. They define α_k to be proportional to the cosine similarity between the target unseen word (t) and neighbor (n_k):

$$\alpha_k = s(t, n_k) = \frac{\phi_o(t).\phi_o(n_k)}{|\phi(t)||\phi_o(n_k)|}$$

	Avg. UAS	Avg. LAS
Baseline	84.11	81.02
k-NN	84.54	81.38
Our Mapped	84.97	81.79

Table 6.4: Comparison to k-nearest neighbor matching of Tafforeau et al. (2015).

Table 6.4 shows the average performance of the models over the development sets of the Web Treebank. On average, our mapper outperforms the k-NN approach ($k = 3$).

6.4.7 *Dependency Parsing Examples*

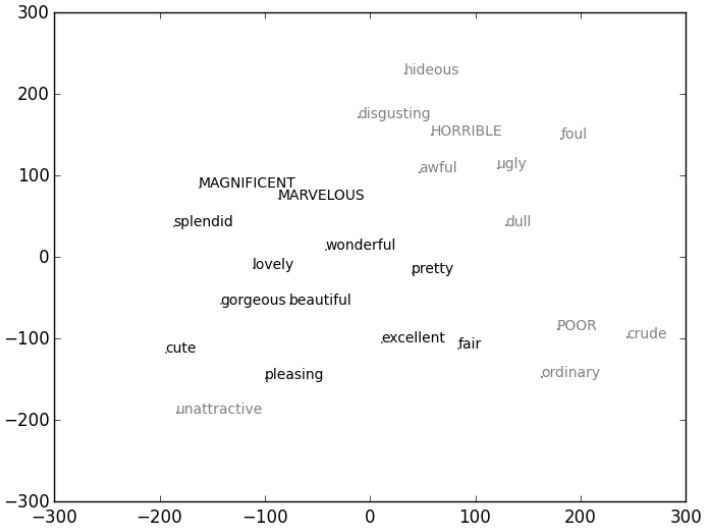
In Figure 6.2, we show two sentences: an instance where the mapper helps and another where the mapper hurts the parsing performance.[4] In the first sentence (Figure 6.2a), the parsing model has not seen the word 'attempted' during training. Note that the sentence contains 3 verbs: 'attempted', 'adopt', and 'was'. Even with the POS tags, the parser was unable to get the correct dependency attachment. After mapping, the parser correctly makes 'attempted' the root and gets the correct arcs and the correct tree. The 3 nearest neighbors of 'attempted' in the mapped embedding space are 'attempting', 'tried', and 'attempt'. We also see here that a single unseen word can lead to multiple errors in the parse.

In the second example (Figure 6.2b), the default model assigns the correct arcs using the POS information even though it has not seen the word 'google'. However, using the mapped representation for 'google', the parser makes errors. The 3-nearest neighbors for 'google' in the mapped space are 'damned', 'look', and 'hash'. We hypothesize that the mapper has mapped this noun instance of 'google' to be closer to verbs instead of nouns, which would explain the incorrect attachment.

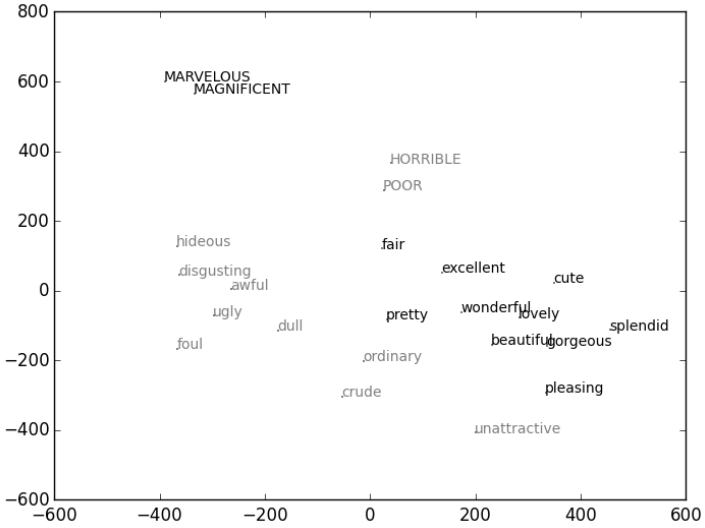
6.4.8 *Analyzing Mapped Representations*

To understand the mapped embedding space, we use t-SNE [Van der Maaten and Hinton, 2008] to visualize a small subset of embeddings. In Figure 6.3, we plot the initial embeddings, the parser-trained embeddings, and finally the mapped embeddings. We include four unseen words (shown in caps): 'horrible', 'poor', 'marvelous', and 'magnificent'. In Figure 6.3a and Figure 6.3b, the embeddings for the unseen words are identical (even though t-SNE places them in different places when producing its projection). In Figure 6.3c, we observe that the mapper has placed the unseen words within appropriate areas of the space with respect to similarity with the seen words. We contrast this with Figure 6.3b, in which the unseen words appear to be within a different region of the space from all seen words.

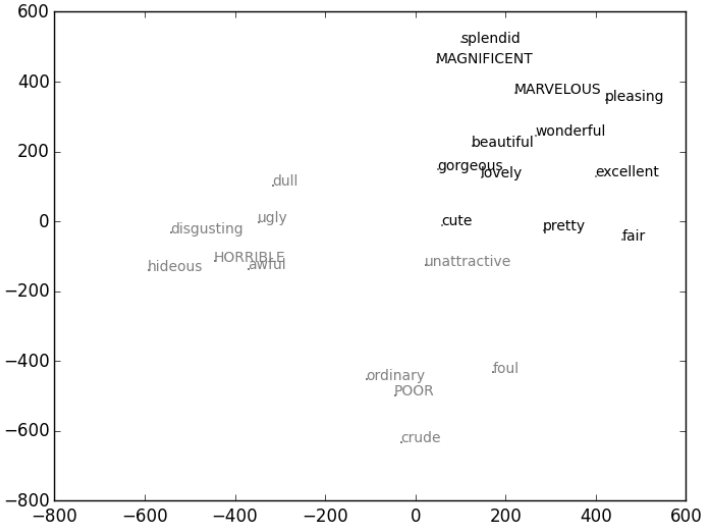
4 Sentences in Figure 6.2 are taken from the development portion of the Answers domain from the Web Treebank.



(a) Initial Representational space



(b) Learned Representational Space



(c) Mapped Representational Space

Figure 6.3: t-SNE plots on initial, parser trained, and mapped representations.

6.5 SUMMARY

We have described a simple method to resolve unseen words when training supervised models that learn task-specific word embeddings: a feed-forward neural network that maps initial embeddings to the task-specific embedding space. We demonstrated significant improvements in dependency parsing accuracy across several domains, as well as improvements on a downstream task. Our approach is simple, effective, and applicable to many other settings, both inside and outside NLP.

7 CONCLUSION AND FUTURE DIRECTIONS

7.1 SUMMARY AND CONCLUSIONS

Our thesis focuses on defining methods that make use of lexical representations, which are learned from large data sources. We observe that our proposed methods are able to generalize to words that are not seen during training.

Specifically, we describe methods to tailor word embeddings such that they are used for specific tasks in NLP in an efficient manner. We propose probabilistic methods that use the unsupervised representations as initial features for a downstream task, such as prepositional phrase attachment, and show that these models complement the linear models. We expanded from using bilinear models to multilinear models and observed that these simple yet effective models outperform neural network based models under the same setting.

Our framework also performs representational compression such that the word embeddings are compressed in a low-dimensional space by using nuclear-norm regularization. We obtain high-quality task specified embeddings that are computationally efficient for downstream tasks. We also use a similar model setting to map embeddings from two different languages to a common subspace and effectively improve the performance on machine translation tasks, especially for out of domain dataset. These empirical results encourage the application of our modeling framework to interesting areas of research in natural language parsing and related tasks.

One of the common themes in this thesis has been about the use of nuclear-norm regularization that gives us low-rank models. We have seen throughout our work that low-rank based models are amongst the best performing models and have several advantages including a) they lead to superior models with fewer parameters and exploit latent interactions, especially in cases where the initial word representations are sparse, and b) they lead to a relatively lean and simple way to obtain task specific compressions over word embeddings.

Further, we describe a non-linear mapping function that maps initial representations to task-specified representations. We used a novel weighted multi-loss function, which can further be tuned for domain adaptation using a tiny amount of supervision. This method is especially useful when we have supervised data in one domain (such as newswire), but we have to parse more real world domains (for example emails, etc.,) where there is a large vocabulary mismatch.

7.2 FUTURE DIRECTIONS

Some of the possible directions for future research include:

7.2.1 *Other NLP Tasks and Languages*

We would like to try our approaches in other advanced natural language processing tasks that include structured prediction tasks. Typically, a structured prediction model in NLP uses some form of a composition of lexical units, utilizing our proposed binary, ternary, or n-ary compositions using low-rank regularization could likely be a neat fit.

This thesis has mostly focussed on different tasks for English. An immediate extension of the work presented in the thesis is an evaluation on different languages. Our methods are especially useful for low-resource languages. In some cases, where we perform experiments on other languages, like Arabic and Spanish, we see similar improvements as in English. We are also interested in experimenting with transfer learning between languages. As some languages have inter lingual syntacto-semantic similarity, it would be interesting to study if bilinear model characteristics can be transferred between languages.

7.2.2 *Multimodal Task Specific Representations*

Multimodal tasks are tasks that are conditioned on two different modalities eg., models conditioned on both vision based features and text based features, models conditioned on both speech based features and text based features, etc.. Recent work has shown that bilinear models are advantageous in the context of multimodal tasks. Previous work [Fukui et al., 2016; Kim et al., 2017; Lin et al., 2015; Pirsiavash et al., 2009] has demonstrated the utility of simple bilinear models in the context of vision and language, especially some of the these have achieved state-of-the-results in the tasks. We would like to apply our bilinear low-rank models in the context of multimodal tasks.

We have already approached this problem in the context of obtaining semantic roles given an image [Quattoni et al., 2016]. With more sophisticated model architectures, bilinear models that are able to make use of end-to-end learning could yield interesting results and insights.

7.2.3 *Low-Rank plus Additive Matrices*

With low-rank plus additive matrices regularization, the additive components have complementary structures; for example, in the case of

the sum of a low-rank matrix with a sparse matrix, the low-rank matrices can be seen as the functions that compute the inner product in the low-dimensional space and the sparse matrices learn task related idiosyncrasies and exceptions.

Recent work [Hutchinson et al., 2013, 2015; Candès et al., 2011] demonstrates the usefulness of low-rank plus additive matrices. Our proposed models in the thesis can be naturally extended to have low-rank plus sparse components.

BIBLIOGRAPHY

E. Agirre, T. Baldwin, and D. Martinez. Improving parsing and pp attachment performance with sense information. In *Association Computational Linguistics*, pages 317–325, 2008.

E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics, 2009.

Y. Al-Onaizan and K. Knight. Machine transliteration of names in arabic text. In *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages*, pages 1–13. Association for Computational Linguistics, 2002.

A. J. Anderson, E. Bruni, U. Bordignon, M. Poesio, and M. Baroni. Of words, eyes and brains: Correlating image-based distributional semantic models with neural representations of concepts. In *Empirical Methods in Natural Language Processing*, pages 1960–1970. Association of Computational Linguistics, 2013.

J. Andreas and D. Klein. How much do word embeddings encode about syntax? In *Proceedings of Annual Meeting of Association of Computational Linguistics*, Baltimore, Maryland, USA, June 2014.

S. C. AP, S. Lauly, H. Larochelle, M. Khapra, B. Ravindran, V. C. Raykar, and A. Saha. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*, pages 1853–1861, 2014.

S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4, 2016.

R. F. Astudillo, S. Amir, W. Lin, M. Silva, and I. Trancoso. Learning word representations from scarce and noisy data with embedding sub-spaces. *Proceedings of the Association for Computational Linguistics (ACL), Beijing, China*, 2015.

M. Attia, J. Foster, D. Hogan, J. L. Roux, L. Tounsi, and J. Van Genabith. Handling unknown words in statistical latent-variable parsing models for arabic, english and french. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of*

Morphologically-Rich Languages, pages 67–75. Association for Computational Linguistics, 2010.

F. R. Bach. Consistency of the group lasso and multiple kernel learning. *The Journal of Machine Learning Research*, 9:1179–1225, 2008.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*, 2015.

B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, O. Chapelle, and K. Weinberger. Learning to rank with (a lot of) word features. *Information Retrieval*, 13(3):291–314, June 2010. ISSN 1386-4564. doi: 10.1007/s10791-009-9117-9. URL http://dx.doi.org/10.1007/s10791-009-9117-9.

L. D. Baker and A. K. McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98', pages 96–103, New York, NY, USA, 1998. ACM. ISBN 1-58113-015-5. doi: 10.1145/290941.290970. URL http://doi.acm.org/10.1145/290941.290970.

B. Balle and M. Mohri. Spectral learning of general weighted automata via constrained matrix completion. *Neural Information Processing Systems Conference (NIPS)*, 2012.

M. Ballesteros, C. Dyer, and N. A. Smith. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/D15-1041.

S. Banerjee and A. Lavie. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

M. Bansal. Dependency link embeddings: Continuous representations of syntactic substructures. In *Proceedings of the NAACL Workshop on Vector Space Modeling for NLP*, 2015.

M. Bansal and D. Klein. Web-scale features for full-scale parsing. In *Proceedings of Annual Meeting of Association of Computational Linguistics*, 2011.

M. Bansal, K. Gimpel, and K. Livescu. Tailoring continuous word representations for dependency parsing. In *Proceedings of Annual Meeting of Association of Computational Linguistics*, 2014a.

M. Bansal, K. Gimpel, and K. Livescu. Tailoring continuous word representations for dependency parsing. In *Proceedings of Annual Meeting of Association of Computational Linguistics*, 2014b.

M. Baroni. Distributional semantics with eyes. In *Saarbruecken Colloquium*. 2012.

M. Baroni and A. Lenci. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4): 673–721, 2010.

M. Baroni, G. Dinu, and G. Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 238–247, 2014.

Y. Belinkov, T. Lei, R. Barzilay, and A. Globerson. Exploring compositional architectures and word vector representations for prepositional phrase attachment. *Transactions of the Association for Computational Linguistics*, 2:561–572, 2014.

Y. Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3: 1137–1155, 2003.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

D. M. Bikel. A distributional analysis of a lexicalized statistical parsing model. In D. Lin and D. Wu, editors, *Proceedings of Empirical Methods in Natural Language Processing 2004*, pages 182–189, Barcelona, Spain, July 2004. Association for Computational Linguistics.

D. Blei and J. McAuliffe. Supervised topic models. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 121–128. MIT Press, Cambridge, MA, 2008.

J. A. Botha and P. Blunsom. Compositional morphology for word representations and language modelling. In *International Conference on Machine Learning (ICML)*, 2014.

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

E. Bruni, N.-K. Tran, and M. Baroni. Multimodal distributional semantics. *Journal of Artificial Intelligence Research (JAIR)*, 49:1–47, 2014.

J. A. Bullinaria and J. P. Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*, 39(3):510–526, 2007.

J. A. Bullinaria and J. P. Levy. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and svd. *Behavior research methods*, 44(3):890–907, 2012.

S. Calhoun, J. Carletta, J. M. Brenier, N. Mayo, D. Jurafsky, M. Steedman, and D. Beaver. The NXT-format Switchboard Corpus: A rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue. *Language resources and evaluation*, 44(4):387–419, Dec. 2010. ISSN 1574-020X. doi: 10.1007/s10579-010-9120-1. URL http://dx.doi.org/10.1007/s10579-010-9120-1.

E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

M. Candito and B. Crabbé. Improving generative statistical parsing with semi-supervised word clustering. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 138–141. Association for Computational Linguistics, 2009.

E. Charniak and M. Johnson. Edit detection and parsing for transcribed speech. In *Proceedings of the Second Conference of the North American chapter of the Association for Computational Linguistics (NAACL '01)*, 2001. URL http://acl.ldc.upenn.edu/N/N01/N01-1016.pdf.

E. Charniak, D. Blaheta, N. Ge, K. Hall, and M. Johnson. BLLIP 1987–89 WSJ Corpus Release 1, LDC No. LDC2000T43. Linguistic Data Consortium, 2000.

G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, pages 1109–1135, 2010. URL http://jmlr.csail.mit.edu/papers/v11/chechik10a.html.

D. Chen and C. D. Manning. A fast and accurate dependency parser using neural networks. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014a.

D. Chen and C. D. Manning. A fast and accurate dependency parser using neural networks. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014b.

Y. Chen, B. Perozzi, and S. Skiena. Vector-based similarity measurements for historical figures. In *SISAP*, volume 9371 of *Lecture Notes in Computer Science*, pages 179–190. Springer, 2015.

J. Cheng and D. Kartsaklis. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1531–1542, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/D15-1177.

R. Chitnis and J. DeNero. Variable-length word encodings for neural translation models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2088–2093, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/D15-1249.

K. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

V. Cirik and H. Sensoy. The ai-ku system at the spmrl 2013 shared task: Unsupervised features for dependency parsing. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 68–75, 2013.

R. Cohen, Y. Goldberg, and M. Elhadad. Domain adaptation of a dependency parser with a class-class selectional preference model. In *Proceedings of ACL 2012 Student Research Workshop*, pages 43–48. Association for Computational Linguistics, 2012.

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

R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

R. Collobert, W. Jason, B. Leon, K. Michael, K. Koray, and K. Pavel. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011a.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:24932537, 2011b.

M. R. Costa-jussà, C. E. a Bonet, P. Madhyastha, C. Escolano, and J. A. R. Fonollosa. The talp-upc spanish-english wmt biomedical task: Bilingual embeddings and char-based neural language model

rescoring in a phrase-based system. In *Proceedings of First Conference on Machine Translation (WMT)*, 2016.

H. Daumé III and J. Jagarlamudi. Domain adaptation for machine translation by mining unseen words. In *Association for Computational Linguistics*, Portland, OR, 2011. URL http://hal3.name/docs/#daume11lexicaladapt.

M.-C. De Marneffe and C. D. Manning. Stanford typed dependencies manual. Technical report, Stanford University, 2008.

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

J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, and J. Makhoul. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1370–1380, 2014.

P. S. Dhillon, D. P. Foster, and L. H. Ungar. Eigenwords: Spectral word embeddings. *The Journal of Machine Learning Research*, 2015.

G. Dinu, A. Lazaridou, and M. Baroni. Improving zero-shot learning by mitigating the hubness problem. *In proceedings of International Conference on Representation Learning - Workshop Track*, 2015.

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

N. Durrani and P. Koehn. Improving machine translation via triangulation and transliteration. In *Proceedings of the 17th Annual Conference of the European Association for Machine Translation (EAMT), Dubrovnik, Croatia*, 2014.

C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith. 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)*, pages 334–343, Beijing, China, July 2015a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1033.

C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith. 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)*, pages 334–343, Beijing, China, July 2015b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1033.

B. Efron and R. Tibshirani. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science*, 1(1):54–75, 02 1986. doi: 10.1214/ss/1177013815. URL http://dx.doi.org/10.1214/ss/1177013815.

L. Ellebracht, A. Ramisa, P. Swaroop, J. Cordero-Rama, F. Moreno-Noguer, and A. Quattoni. Semantic tuples for evaluation of image sentence generation. In *Vision and Language Workshop (in EMNLP)*, 2015.

M. Faruqui and C. Dyer. Improving vector space word representations using multilingual correlation. In *In Proceedings of 44th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2014.

M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith. Retrofitting word vectors to semantic lexicons. *In Proceedings of North American Association of Computational Linguistics*, 2015a.

M. Faruqui, Y. Tsvetkov, D. Yogatama, C. Dyer, and N. Smith. Sparse overcomplete word vector representations. *In Proceedings of Annual Meeting of Association of Computational Linguistics*, 2015b.

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM, 2001.

J. R. Firth. A synopsis of linguistic theory 1930-55. *Studies in Linguistic Analysis (special volume of the Philological Society)*, 1952-59:1–32, 1957.

M. L. Forcada, M. Ginestí-Rosell, J. Nordfalk, J. OŔegan, S. Ortiz-Rojas, J. A. Pérez-Ortiz, F. Sánchez-Martínez, G. Ramírez-Sánchez, and F. M. Tyers. Apertium: a free/open-source platform for rule-based machine translation. *Machine translation*, 25(2):127–144, 2011.

J. Foster. cba to check the spelling investigating parser performance on discussion forum posts. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 381–384. Association for Computational Linguistics, 2010.

J. Foster, Ö. Çetinoglu, J. Wagner, J. Le Roux, S. Hogan, J. Nivre, D. Hogan, and J. Van Genabith. Pos tagging and parsing the twitterverse. In *AAAI 2011 Workshop on Analyzing Microtext*, pages 20–25, 2011.

A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, pages 2121–2129, 2013.

A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. *Preprint at Computing Research Repository; arXiv:1606.01847*, 2016.

J. Gao, X. He, W.-t. Yih, and L. Deng. Learning semantic representations for the phrase translation model. *TechReport preprint arXiv:1312.0482*, 2013.

S. Gouws and A. Søgaard. Simple task-specific bilingual word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1386–1390, Denver, Colorado, May–June 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N15-1157.

S. Gouws, Y. Bengio, and G. Corrado. BilBOWA: Fast Bilingual Distributed Representations without Word Alignments. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 748–756, July 2015.

G. Grefenstette. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Norwell, MA, USA, 1994. ISBN 0792394682.

D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks. A closer look at skip-gram modelling. In *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*, pages 1–4, 2006.

N. Habash. Remoov: A tool for online handling of out-of-vocabulary words in machine translation. In *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR), Cairo, Egypt*, 2009.

G. Halawi, G. Dror, E. Gabrilovich, and Y. Koren. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414. ACM, 2012.

Z. Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

J. Hauke and T. Kossowski. Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data. *Quaestiones Geographicae*, 30(2):87–93, 2011.

K. M. Hermann and P. Blunsom. Multilingual models for compositional distributed semantics. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.

U. Hermjakob, K. Knight, and H. Daumé III. Name translation in statistical machine translation-learning when to transliterate. In *In*

Proceedings of Annual Meeting of Association of Computational Linguistics, pages 389–397, 2008.

F. Hill and A. Korhonen. Learning abstract concept embeddings from multi-modal data: Since you probably can't see what i mean. In *Empirical Methods in Natural Language Processing*, pages 255–265, 2014.

F. Hill, R. Reichart, and A. Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 2015.

D. Hindle. Noun classification from predicate-argument structures. In *Proceedings of the 28th Annual Meeting on Association for Computational Linguistics*, Association of Computational Linguistics '90', pages 268–275, Stroudsburg, PA, USA, 1990. Association for Computational Linguistics. doi: 10.3115/981823.981857. URL http://dx.doi.org/10.3115/981823.981857.

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

G. E. Hinton. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.

M. Honnibal and M. Johnson. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics*, 2:131–142, 2014.

H. Hotelling. Relations between two sets of variates. *Biometrika*, 28 (3/4):321–377, 1936.

E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. 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*, pages 873–882. Association for Computational Linguistics, 2012a.

E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. 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*, pages 873–882. Association for Computational Linguistics, 2012b.

F. Huang and A. Yates. Distributional representations for handling sparsity in supervised sequence labeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2009.

F. Huang and A. Yates. Exploring representation-learning approaches to domain adaptation. In *Proceedings of the Association of Computational Linguistics 2010 Workshop on Domain Adaptation for Natural Language Processing (DANLP)*, 2010.

F. Huang, A. Ahuja, D. Downey, Y. Yang, Y. Guo, and A. Yates. Learning Representations for Weakly Supervised Natural Language Processing Tasks. *Computational Linguistics*, xx:yy, 2013.

Z. Huang and M. Harper. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 832–841, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D/D09/D09-1087.

Z. Huang and M. P. Harper. Feature-rich log-linear lexical model for latent variable PCFG grammars. In *Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011, Chiang Mai, Thailand, November 8-13, 2011*, pages 219–227, 2011. URL http://aclweb.org/anthology/I/I11/I11-1025.pdf.

B. Hutchinson, M. Ostendorf, and M. Fazel. Exceptions in language as learned by the multi-factor sparse plus low-rank language model. In *ICASSP*, pages 8580–8584, 2013.

B. Hutchinson, M. Ostendorf, and M. Fazel. A sparse plus low-rank exponential language model for limited resource scenarios. *IEEE Transactions on Audio, Speech, and Language Processing*, 23(3):494–504, 2015.

S. Ishiwatari, N. Yoshinaga, M. Toyoda, and M. Kitsuregawa. Instant translation model adaptation by translating unseen words in continuous vector space. *Conference on Intelligent Text Processing and Computational Linguistics*, 2016.

S. Jean, K. Cho, R. Memisevic, and Y. Bengio. On using very large target vocabulary for neural machine translation. 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)*, pages 1–10, Beijing, China, July 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1001.

B. John, M. Ryan, and P. Fernando. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia, 2006.

D. A. Jurgens, P. D. Turney, S. M. Mohammad, and K. J. Holyoak. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the First Joint Conference on Lexical and Computational*

Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation, pages 356–364. Association for Computational Linguistics, 2012.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P14-1062.

F. Keller and M. Lapata. Using the web to obtain frequencies for unseen bigrams. *Computational linguistics*, 29(3):459–484, 2003.

J.-H. Kim, K.-W. On, J. Kim, J.-W. Ha, and B.-T. Zhang. Hadamard product for low-rank bilinear pooling. *In Proceedings of International Conference on Learning Representations*, 2017.

Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1181.

Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush. Character-aware neural language models. *In Proceedings of Association for the Advancement of Artificial Intelligence*, 2016. URL http://arxiv.org/abs/1508.06615.

R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler. Skip-thought vectors. *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2015.

A. Klementiev, I. Titov, and B. Bhattarai. Inducing crosslingual distributed representations of words. 2012.

T. Kočiskỳ, K. M. Hermann, and P. Blunsom. Learning bilingual word representations by marginalizing alignments. *In Proceedings of Annual Meeting of Association of Computational Linguistics*, 2014.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the Associtation of Computational Linguistics on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics, 2007a.

P. Koehn, H. Hoang, A. B. Mayne, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for

statistical machine translation. In *Annual Meeting of the Association for Computation Linguistics (ACL), Demonstration Session*, pages 177–180, June 2007b. URL http://www.iccs.inf.ed.ac.uk/~pkoehn/publications/acl2007-moses.pdf.

A. Köhn, U. C. Lao, A. B. Zadeh, and K. Sagae. Parsing morphologically rich languages with (mostly) off-the-shelf software and word vectors. *SPMRL-SANCL 2014*, 2014.

L. Kong, N. Schneider, S. Swayamdipta, A. Bhatia, C. Dyer, and N. A. Smith. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1001–1012, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1108.

T. Koo, X. Carreras, and M. Collins. Simple semi-supervised dependency parsing. In *Proceedings of Annual Meeting of Association of Computational Linguistics*, pages 595–603, Columbus, Ohio, June 2008. Association for Computational Linguistics.

J. K. Kummerfeld, D. Hall, J. R. Curran, and D. Klein. 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*, pages 1048–1059. Association for Computational Linguistics, 2012.

M. Labeau, K. Löser, and A. Allauzen. Non-lexical neural architecture for fine-grained pos tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 232–237, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/D15-1025.

I. Labutov and H. Lipson. Re-embedding words. In *In Proceedings of Annual Meeting of Association of Computational Linguistics (2)*, pages 489–493, 2013.

T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

G. Lapesa and S. Evert. A large scale evaluation of distributional semantic models: Parameters, interactions and model selection. *Transactions of the Association for Computational Linguistics*, 2:531–545, 2014.

A. Lazaridou, G. Dinu, and M. Baroni. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *In Proceedings of Annual Meeting of Association of Computational Linguisitcs*, pages 270–280, 2015.

C. Leacock, G. Towell, and E. Voorhees. Corpus-based statistical sense resolution. In *Proceedings of the workshop on Human Language Technology*, pages 260–265. Association for Computational Linguistics, 1993.

R. Lebret and R. Collobert. Word emdeddings through hellinger pca. *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, 2014.

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

O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014a.

O. Levy and Y. Goldberg. Dependencybased word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308, 2014b.

P. Liang. *Semi-supervised learning for natural language*. PhD thesis, Massachusetts Institute of Technology, 2005.

D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 768–774, Montreal, Quebec, Canada, August 1998. Association for Computational Linguistics.

D. Lin and P. Pantel. Dirt @sbt@discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01', pages 323–328, New York, NY, USA, 2001. ACM. ISBN 1-58113-391-X.

D. Lin and X. Wu. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1030–1038, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P09/P09-1116.

T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1457, 2015.

W. Ling, T. Luís, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. W. Black, and I. Trancoso. Finding function in form: Compositional

character models for open vocabulary word representation. In *Proc. of Empirical Methods in Natural Language Processing*, 2015.

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3, (Ser. B)): 503–528, 1989.

K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods Instruments and Computers*, 28(2):203–208, 1996. URL http://scholar.google.de/scholar.bib?q=info:BfG544ylGnkJ:scholar.google.com/&output=citation&hl=de&as_sdt=2000&as_vis=1&ct=citation&cd=0.

M.-T. Luong, R. Socher, and C. D. Manning. Better word representations with recursive neural networks for morphology. *Conference on Computational Natural Language Learning*, 104, 2013a.

T. Luong, R. Socher, and C. Manning. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria, August 2013b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-3512.

T. Luong, I. Sutskever, Q. Le, O. Vinyals, and W. Zaremba. Addressing the rare word problem in neural machine translation. 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)*, pages 11–19, Beijing, China, July 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1002.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

P. S. Madhyastha and C. España-Bonet. Resolving out-of-vocabulary words with bilingual embeddings in machine translation. *Preprint in Computational Research Repository; arXiv:1608.01910*, 2016.

P. S. Madhyastha, X. Carreras, and A. Quattoni. Tailoring word embeddings for bilexical predictions: An experimental comparison. *International Conference on Learning Representations 2015, Workshop Track*, 2015.

P. S. Madhyastha, M. Bansal, K. Gimpel, and K. Livescu. Mapping unseen words to task-trained embedding spaces. *1st Workshop on*

Representational Learning for NLP, In Proceedings of Annual Meeting of Association of Computational Linguistics, 2016.

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

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993a.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993b.

S. Martin, J. Liermann, and H. Ney. Algorithms for bigram and trigram word clustering. *Speech communication*, 24(1):19–37, 1998.

A. Martins, M. Almeida, and A. N. Smith. 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)*, pages 617–622. Association for Computational Linguistics, 2013. URL http://aclweb.org/anthology/P13-2109.

Y. Marton, N. Habash, and O. Rambow. Improving arabic dependency parsing with lexical and inflectional morphological features. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 13–21, Los Angeles, CA, USA, June 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-1402.

D. McClosky, E. Charniak, and M. Johnson. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 337–344. Association for Computational Linguistics, 2006.

D. McClosky, E. Charniak, and M. Johnson. Automatic domain adaptation for parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 28–36, Los Angeles, California, June 2010. Association for Computational Linguistics.

R. T. McDonald and J. Nivre. Characterizing the errors of data-driven dependency parsing models. In *Empirical Methods in Natural Language Processing*, pages 122–131, 2007.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at ICLR*, 2013b.

T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting Similarities among Languages for Machine Translation. *Preprint on Computing Research Repository*, abs/1309.4168, 2013c.

T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013d.

G. A. Miller and W. G. Charles. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28, 1991.

S. Miller, J. Guinness, and A. Zamanian. Name tagging with word clusters and discriminative training. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 4, pages 337–342, 2004a.

S. Miller, J. Guinness, and A. Zamanian. Name tagging with word clusters and discriminative training. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 4, pages 337–342, 2004b.

A. Mnih, Z. Yuecheng, and G. Hinton. Improving a statistical language model through non-linear prediction. *Neurocomputing*, 72 (7-9):1414–1418, 2009.

F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252, 2005.

B. Murphy, P. P. Talukdar, and T. M. Mitchell. Learning effective and interpretable semantic models using non-negative sparse embedding. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pages 1933–1950, 2012.

N. Nakashole and T. M. Mitchell. 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, 2015.

S. Necsulescu, S. Mendes, D. Jurgens, N. Bel, and R. Navigli. Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 182–192, Denver, Colorado, June 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/S15-1021.

A. Neelakantan, J. Shankar, A. Passos, and A. McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. *In Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2014.

J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, Q. V. Le, and A. Y. Ng. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 265–272, 2011.

J. Nilsson, S. Riedel, and D. Yuret. The conll 2007 shared task on dependency parsing. In *Proceedings of the CoNLL shared task session of Conference on Computational Natural Language Learning*, pages 915–932. sn, 2007.

F. J. Och. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July 6-7 2003.

F. J. Och and H. Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302. Association for Computational Linguistics, 2002.

F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. 29(1):19–51, 2003.

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

S. Pado and M. Lapata. Cross-lingual projection of role-semantic information. In *Proceedings of Empirical Methods in Natural Language Processing 2005*, Vancouver, BC, 2005.

S. Pado and M. Lapata. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199, 2007.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the Association of Computational Linguistics*, pages 311–318, 2002.

R. Paulus, R. Socher, and C. D. Manning. Global belief recursive neural networks. In *Advances in Neural Information Processing Systems*, pages 2888–2896, 2014.

J. Pennington, R. Socher, and D. M. Christopher. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543, 2014a.

J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1162.

F. Pereira, N. Tishby, and L. Lee. Distributional clustering of english words. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 183–190. Association for Computational Linguistics, 1993.

S. Petrov and R. McDonald. Overview of the 2012 shared task on parsing the web, 2012a.

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

S. Petrov, P.-C. Chang, M. Ringgaard, and H. Alshawi. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 705–713. Association for Computational Linguistics, 2010.

H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Bilinear classifiers for visual recognition. In *Advances in neural information processing systems*, pages 1482–1490, 2009.

A. Primadhanty, X. Carreras, and A. Quattoni. 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)*, pages 126–135, Beijing, China, July 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1013.

L. Qu, G. Ferraro, L. Zhou, W. Hou, N. Schneider, and T. Baldwin. Big data small data, in domain out-of domain, known word unknown word: The impact of word representation on sequence labelling tasks. *Proceedings of the 19th Conference on Computational Language Learning*, 2015a.

L. Qu, G. Ferraro, L. Zhou, W. Hou, N. Schneider, and T. Baldwin. Big data small data, in domain out-of domain, known word unknown word: The impact of word representation on sequence labelling tasks. *Proceedings of the 19th Conference on Computational Language Learning*, 2015b.

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

A. Quattoni, A. Ramisa, P. S. Madhyastha, E. Simo-Serra, and F. Moreno-Noguer. Structured prediction with output embeddings for semantic image annotation. *In Proceedings of North American Conference on Computational Linguistics*, Sep 2016.

K. Radinsky, E. Agichtein, E. Gabrilovich, and S. Markovitch. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM, 2011.

R. Rapp. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 519–526. Association for Computational Linguistics, 1999.

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

A. Ratnaparkhi, J. Reynar, and S. Roukos. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the workshop on Human Language Technology*, HLT '94, pages 250–255, Stroudsburg, PA, USA, 1994b. Association for Computational Linguistics. ISBN 1-55860-357-3. doi: 10.3115/1075812.1075868. URL http://dx.doi.org/10.3115/1075812.1075868.

J. Reisinger and R. J. Mooney. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics, 2010.

D. L. Rohde, L. M. Gonnerman, and D. C. Plaut. An improved model of semantic similarity based on lexical co-occurrence. *Communications of the ACM*, 8:627–633, 2006.

S. Rothe and H. Schütze. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2015.

H. Rubenstein and J. B. Goodenough. Contextual correlates of synonymy. *Communications of Annual Meeting of Association for Computing Machinery*, 8(10):627–633, Oct. 1965. ISSN 0001-0782. doi: 10.1145/365628.365657. URL http://doi.acm.org/10.1145/365628.365657.

M. Sahlgren. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. PhD thesis, Stockholm University, 2006.

G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, Aug. 1988. ISSN 0306-4573.

T. Schnabel, I. Labutov, D. Mimno, and T. Joachims. Evaluation methods for unsupervised word embeddings. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

H. Schütze and J. O. Pedersen. Information retrieval based on word senses. In *Fourth Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, NV, 1995.

D. Seddah, G. Chrupała, O. Cetinoglu, J. van Genabith, and M. Candito. Lemmatization and lexicalized statistical parsing of morphologically-rich languages: the case of french. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 85–93, Los Angeles, CA, USA, June 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-1410.

T. Shi, Z. Liu, Y. Liu, and M. Sun. Learning cross-lingual word embeddings via matrix co-factorization. In *In Proceedings of Annual Meeting of Association of Computational Linguistics (2)*, pages 567–572, 2015.

S. Singh, T. Rocktaschel, and S. Riedel. Towards combined matrix and tensor factorization for universal schema relation extraction. In *NAACL Workshop on Vector Space Modeling for NLP (VSM)*, June 2015.

J. R. Smith, C. Quirk, and K. Toutanova. Extracting Parallel Sentences from Comparable Corpora Using Document Level Alignment. In *In Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Cha pter of the Association for Computational Linguistics (NAACL-HLT 2010)*, pages 403–411, 2010.

N. A. Smith and J. Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362. Association for Computational Linguistics, 2005.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the Seventh Conference of the Association for Machine Translation in the Americas (AMTA 2006)*, pages 223–231, Cambridge, Massachusetts, USA, 2006.

R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011.

R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013a.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D13-1170.

R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642, 2013c.

A. Søgaard and A. Johannsen. Robust learning in random subspaces: Equipping NLP for OOV effects. In *Proceedings of COLING 2012: Posters*, Mumbai, India, December 2012.

N. Srebro, J. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1329–1336, 2004.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

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

A. Stolcke. SRILM - An Extensible Language Modeling Toolkit. In *Proceedings of the Seventh International Conference of Spoken Language Processing (ICSLP2002)*, pages 901–904, Denver, Colorado, USA, 2002. URL citeseer.ist.psu.edu/stolcke02srilm.html.

K. Stratos, D.-k. Kim, M. Collins, and D. Hsu. A spectral algorithm for learning class-based n-gram models of natural language. *Proceedings of the Association for Uncertainty in Artificial Intelligence*, 2014.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

J. Tafforeau, T. Artieres, B. Favre, and F. Bechet. Adapting lexical representation and oov handling from written to spoken language with word embedding. In *Interspeech*, 2015.

K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2015.

D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1555–1565, 2014.

F. Tian, H. Dai, J. Bian, B. Gao, R. Zhang, E. Chen, and T.-Y. Liu. A probabilistic model for learning multi-prototype word embeddings. In *Proceedings of COLING*, pages 151–160, 2014.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

J. Turian, L. Ratinov, and Y. Bengio. 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, 2010a.

J. Turian, L. Ratinov, and Y. Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA, 2010b. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1858681.1858721.

P. D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of the 12th European Conference on Machine Learning*, EMCL '01', pages 491–502, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42536-5. URL http://dl.acm.org/citation.cfm?id=645328.650004.

P. D. Turney. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416, 2006.

P. D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1): 141–188, Jan. 2010. ISSN 1076-9757.

S. Upadhyay, M. Faruqui, C. Dyer, and D. Roth. Cross-lingual models of word embeddings: An empirical comparison. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

A. Ushioda and J. Kawasaki. Hierarchical clustering of words and application to nlp tasks. In *Proceedings of the Fourth Workshop on Very Large Corpora*, pages 28–41, 1996.

J. Uszkoreit and T. Brants. Distributed word clustering for large scale class-based language modeling in machine translation. In *In Proceedings of Annual Meeting of Association of Computational Linguistics*, pages 755–762, 2008.

L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

L. Vilnis and A. McCallum. Word representations via gaussian embedding. *In Proceedings of International Conference on Learning Representations*, 2015.

I. Vulic and M.-F. Moens. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, pages 719–725. ACL, 2015.

I. Vulić and M.-F. Moens. Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research*, 55:953–994, 2016.

W. Y. Wang, L. Kong, K. Mazaitis, and W. W. Cohen. Dependency parsing for weibo: An efficient probabilistic logic programming approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1152–1158, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1122.

K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, June 2009. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1577069.1577078.

D. Weiss, C. Alberti, M. Collins, and S. Petrov. Structured training for neural network transition-based parsing. *In Proceedings of Conference on Association for Computational Linguistics*, 2015.

J. Weston, A. Bordes, O. Yakhnenko, and N. Usunier. Connecting language and knowledge bases with embedding models for relation extraction. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013.

P. Wiemer-Hastings and I. Zipitria. Rules for syntax, vectors for semantics. In *In Proceedings of the Twenty-third Annual Conference of the Cognitive Science Society*, pages 1112–1117. Erlbaum, 2001.

H. Wu, D. Dong, X. Hu, D. Yu, W. He, H. Wu, H. Wang, and T. Liu. Improve statistical machine translation with context-sensitive bilingual semantic embedding model. In *Empirical Methods in Natural Language Processing*, pages 142–146, 2014.

M. Xiao and Y. Guo. Distributed word representation learning for cross-lingual dependency parsing. In *CoNLL*, pages 119–129, 2014.

D. Yang and D. M. Powers. *Verb similarity on the taxonomy of WordNet*. 2006.

D. Yogatama and N. Smith. Making the most of bag of words: Sentence regularization with alternating direction method of multipliers. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 656–664, 2014.

D. Yogatama, M. Faruqui, C. Dyer, and N. A. Smith. Learning word representations with hierarchical sparse coding. *Proceedings of the 32 nd International Conference on Machine Learning*, 2015.

M. Yu, M. Dredze, R. Arora, and M. R. Gormley. 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*, pages 1019–1029, San Diego, California, June 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N16-1117.

D. Yuret and M. A. Yatbaz. The noisy channel model for unsupervised word sense disambiguation. *Computational Linguistics*, 36(1):111–127, 2010.

X. Zhang and Y. LeCun. Text understanding from scratch. *Preprint on Computational Research Repository arXiv:1502.01710*, 2015.

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

W. Y. Zou, R. Socher, D. M. Cer, and C. D. Manning. Bilingual word embeddings for phrase-based machine translation. In *Empirical Methods in Natural Language Processing*, pages 1393–1398, 2013.