Analysis of SCODE Word Embeddings Based on Substitute Distributions in Supervised Tasks

by

Volkan Cirik

A Thesis Submitted to the
Graduate School of Sicences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science
in
Computer Engineering

Koç University

August 2014

Koç University

Graduate School of Sciences and Engineering

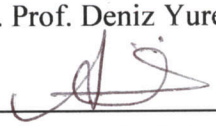
This is to certify that I have examined this copy of a master's thesis by

Volkan Cirik

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

Assoc. Prof. Deniz Yuret (Advisor)



Assistant Prof. Arzucan Özgür

Professor Tunga Güngör

Date: 08.08.2014

Analysis of SCODE Word Embeddings based on Substitute Distributions in Supervised Tasks

Volkan Cirik

August, 2014

To my beloved family

Abstract

One of the interests of the Natural Language Processing (NLP) community is to find representations for lexical items using large amount of unlabeled data. Inducing low-dimensional, continuous, dense word vectors, or word embeddings, have become the principal technique to find representations for words. Word embeddings address the issues of the classical categorical representation of words by capturing syntactic and semantic information of words in the dimensions of a vector. These representations are shown to be successful across NLP tasks including Named Entity Recognition, Part-of-speech Tagging, Parsing, and Semantic Role Labeling.

In this work, I analyze a word embedding method in supervised Natural Language Processing (NLP) tasks. The framework maps words on a sphere such that words co-occurring in similar contexts lie closely. The similarity of contexts is measured by the distribution of substitutes that can fill them. I compared word embeddings, including more recent representations, in Named Entity Recognition (NER), Chunking, and Dependency Parsing. I examine the framework in a multilingual setup as well. The results show that the examined method achieves as good as or better results compared to the other word embeddings. The framework is consistent in improving the baseline systems across languages and achieves state-of-the-art results in multilingual dependency parsing.

Özetçe

Doğal dil işleme ile ilgilenen bilim insanlarının ilgi alanlarından biri de kelimeleri temsil edebilecekleri modeller bulmaktır. Kelime gömüleri her kelimeyi düşük boyutlarda, gerçel değerli, yoğun vektörler olarak gösterir ve en yaygın yöntemlerden biri haline gelmiştir. Kelime gömüleri klasik kategoriksel gösterimlerin yerine bir seçenek olarak görülmektedir. Kelime gömüleri bir kelimenin sözdisizmsel ve anlamsal özelliklerini bir vektörün boyutlarında temsil eder. Bu temsillerin Varlık İsmi Tanımlama, Sözdizimsel Analiz gibi gözetimli doğal dil işleme görevlerinde başarılı olduğu gösterilmiştir.

Bu çalışmada, bir kelime gömü yöntemini gözetimli doğal dil işleme görevlerinde inceliyorum. Yöntem, kelimeleri bir küre uzerinde temsil eder öyle ki aynı bağlamda gözlenen kelimeler bu küre üzerinde yakın bir şekilde konumlandırılır. Bağlamların benzerliği ise o bağlamda gözlenebilecek kelimelerin olasılıksal dağılımları kullanılarak yapılır. Kelime gömülerini Varlık İsmi Tanımlama, Gruplama ve Bağlılık Ayrıştırması görevlerinde karşılastırdım. İncelediğim yöntem en az diğerleri kadar başarılı ya da onlardan daha başarılı sonuşlar almıştır. Yöntem başsarısını çok dilli incelemelerde de sürdürmüştür ve Bağlılık Ayrıştırması görevinde bilinen en iyi sonucları elde etmiştir.

Acknowledgements

I would thank my advisor, Deniz Yuret, for giving me the chance to work with him. He patiently advised me for two years. I am honored to be his student and his friend.

I thank Arzucan Özgür and Tunga Güngör for participating in my thesis committee. I especially thank Tunga Güngör for putting poems below the exams, one of them changed my life. I thank my undergradute advisors Cem Ersoy and Suzan Üsküdarlı for their endless support and encouragement.

I thank the members of Koç AI-Lab and friends from other labs. We learned from each other a lot. I owe each one of you.

I thank my friends for being there whenever I need them.

I thank Burcu for every single moment we had together.

I thank my family for being as awesome as a family could be. Without their unconditional love, I would not have the power to persevere.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

I analyze a word embedding method (Yatbaz et al., 2012), in supervised Natural Language Processing (NLP) tasks. The method represents the context of a word by its probable substitutes. Words with their probable substitutes are fed to a co-occurrence framework (SCODE) (Maron et al., 2010). SCODE maps words on a sphere such that words co-occurring in similar contexts lie closely. These word embeddings, SCODE embeddings, achieve state-of-the-art results in inducing part-of-speech (POS) tags for several languages (Yatbaz et al., 2014). However, their use in supervised setups is not well studied so far. This thesis aims to fill this gap.

Word embeddings represent words with real valued vectors. The dimension of word embeddings are generally small compared to the vocabulary size. They do not suffer from sparsity unlike one-hot representations which have the dimensionality of the vocabulary and a single non-zero entry. In addition, they may help reduce the dependence on hand-designed features which are task and language dependent. The hope is that semantic and syntactic features can be learned from raw data by inducing word embeddings (Mikolov et al., 2013c).

Semi-supervised studies employ features extracted from unlabeled data. Word embeddings provide lexical features for semi-supervised systems.Turian et al. (2010) analyzed the use of several word embeddings in Named Entity Recognition (NER) and Chunking. Word embeddings are used as auxiliary

features in near-state-of-the art solutions. Word embeddings improved results in both tasks compared to baseline systems. Following this study, I report results in Chunking and NER benchmarks for SCODE embeddings. In addition, I examine word embeddings in dependency parsing (Lei et al., 2014). I report multilingual dependency results for SCODE embeddings.

SCODE embeddings based on Substitute Distributions achieve comparable or better results compared to other word embeddings. Multilingual results in dependency parsing also suggest that they are consistent in achieving good results accross different languages. I suggest the use of SCODE embeddings as lexical features in supervised NLP tasks.

The structure of the thesis is as follows:

Chapter 2 details the related work on vector based word representations. I list word embeddings I compare to. I review related studies on SCODE embeddings.

Chapter 3 introduces SCODE word embeddings based on substitue word distributions proposed in (Yatbaz et al., 2012). Each component of the framework is explained. I explain how I induce SCODE word embeddings and how I obtain other word embeddings.

Chapter 4 I introduce supervised NLP tasks and their experimental setups. I report how the NLP tasks are evaluated and the results of experiments.

Chapter 5 concludes the thesis with a summary of my work. I offer future topics to be investigated based on results and insights I obtained in this work.

Chapter 2

Related Work

In this chapter, I review previous studies on word embeddings and SCODE word embeddings. Section 2.1 begins with the discussion of word embeddings. I shortly survey other word representations. In Section 2.2, I summarize related studies using SCODE word embeddings.

2.1 Word Representations

Vector representations of words can be categorized in two approaches. First, I summarize these two common approaches. Then, I briefly explain the methods I evaluated in this thesis.

In the first approach, a word-context frequency matrix is constructed. For instance, the context of a word could be the window of words, grammatical dependencies or documents the word occurs in. The frequency counts are often weighted since the rare occurrences are more informative than the expected ones. The two popular ways of weighting the matrices are the tf-idf (Jones, 1972) and Pointwise Mutual Information (Church and Hanks, 1990). The dimensionality of the frequency matrix is reduced with an algebraic method such as Singular Value Decomposition (SVD). The vector representations of words correspond to the rows of the reduced matrix. For instance, Latent Semantic Analysis (LSA) (Deerwester et al., 1990) forms a frequency matrix for words types and their contexts. The context of a word

is the words in a window. They reduce the dimension of the matrix using SVD. Turney and Pantel (2010) review these models in more detail.

The second popular approach is based on neural networks. These methods address language modeling using Neural Networks. In the input layer, the words in the context are used. Words could be represented as one-hot vectors or real-valued dense vectors. The hidden layer projects words into usually lower dimension vectors. In the output layer, the vector representation of the next word or the probability distribution of words are predicted. Word representations are the by-products of this learning process. They achieve competitive results compared to n-gram and discounting based language models.

Among the previous studies on word vector representations, here, I introduce word embeddings I evaluate in this thesis.

- **C&W:** Collobert and Weston (2008) introduce a Neural Network (NN) architecture that is jointly trained for supervised NLP tasks as well as for language modeling. Word vectors are the by-product of the language modeling step. For each training step, vector representations of n-gram sequences x around a target word[1] are concatenated to $e(x)$. A corrupted version of this sequence $e(\tilde{x})$ is also generated where the target word is replaced with a random word from the vocabulary. A single layer NN scores these sequences. The learning objective is that the score of $e(x)$ is greater than $e(\tilde{x})$ by some margin. The loss function is then $L(x) = max(0, 1 - score(e(x)) + s(e(\tilde{x})))$. This loss is minimized by doing gradient descent over the network parameters and word vectors. The score is not normalized over the vocabulary of words, thus, the scores of sequences do not correspond to probabilities. Bengio et al. (2009) achieves better average log-loss using the same architecture by introducing *curriculum learning*. The basic idea is that the easy-to-learn instances should be introduced to the training algorithm first. Following this idea, they first introduce the most frequent words to the

[1] Collobert and Weston (2008) choose target word in the middle of the sequence, whereas in the studies using the same architecture (Turian et al., 2010) and (Bengio et al., 2009) the target word is at the end of the sequence.

NN, than less frequent ones.

- **HLBL:** Mnih and Hinton (2007) propose a graphical model for language modeling. The model starts with random d dimension word vectors with matrix W of size $|V| \times |d|$ where $d \ll |V|$ and V is the vocabulary. Given a sequence of words, the model predicts a word vector using a linear combination of the preceding word vectors. The similarity between a word's vector and the predicted word vector is calculated by the inner product. The probability distribution of the next word is the normalization of the similarities. The model is trained with gradient descent and W is updated. This method is computationally expensive since the normalization step requires the sum over all words in the vocabulary. Morin and Bengio (2005) suggest using a tree hierarchy for words where each leaf of the tree is a word. Thus, a word can be represented as the sequence of binary decisions on the tree. The probability of observing a word given the word sequence is equal to the probability of making the sequence of binary decisions on a tree hierarchy given word sequence. They use WordNet's sense hierarchy to construct the tree. It is faster more than two orders of magnitude compared to the previous language models, however there is a significant decrease in the performance. Mnih and Hinton (2009) introduce a model to construct the hierarchy from data with bootstrapping. They randomly generate a binary tree. A hierarchical language model is trained using the tree. They use the word vectors of the model to construct the new tree. The new tree is constructed as follows. For each occurrence of a word an expectation is calculated by multiplying the predicted word vector and the word's probability occurring in that position. A word's condensed contexts vector is the sum of these expectations. Then two Gaussian mixtures are fitted to these vectors. The words are partitioned into two groups. The same Gaussian fitting is applied within the groups until a group contains only two words. Their approach significantly increase the performance of the language model of (Mnih and Hinton, 2007) and achieve the best results compared to n-gram based language models.

- **GCA NLM:** Huang et al. (2012) extend (Collobert and Weston, 2008) by adding global context component into the architecture. Word sequence before the target word is defined as the local context. The global context is extracted from the document where the word sequences are observed. The document is represented as an ordered list of word vectors. The global context vector is the weighted average of all word vectors in the document. Similar to (Collobert and Weston, 2008), a two-layer neural network computes the global context score where the input is the global context vector. The final score of the model is the sum of the local and global scores. They argue that the local score preserves the syntactic information and the global score captures the semantic and topic of the document.

- **LR-MVL:** Dhillon et al. (2011) use Canonical Correlation Analysis (CCA) to find a matrix A projecting words into low dimensional real-valued vectors. CCA is similar to the Principal Component Analysis (PCA). PCA finds the orthogonal directions that capture the maximum covariance of a matrix. On the other hand, CCA computes the orthogonal directions (eigenvectors) that capture the maximal correlation between a pair of matrices. The method starts with a random A. For a target word, the words on the left and right context are projected into word vectors using A. The CCA between left and right projections are taken. The resulting model has two set of eigenvectos for left and right context. These eigenvectors are multiplied with the vectors of the context words and produce "hidden state" vectors for the target word. By taking the average of the "hidden state" vectors, the matrix A is updated. The model is also able to generate context-dependent word vectors. After finding the A, left / right context eigenvectors, the left and right projections are concatenated with the target word's vector to form context-dependent word vectors. Dhillon et al. (2012) introduce second CCA step into the method of (Dhillon et al., 2011). The second CCA is taken the between the "hidden state" vectors and the target word's vector. The eigenvectors of the new model is used to update

the matrix A.

- **Skip-Gram NLM:** Mikolov et al. (2013a) introduce Skip-Gram model in which a target word in a context is used to predict the words in the context. Input layer is a one-hot vector in the size of vocabulary. The input layer is projected onto a continuous vector with a hidden layer. The hidden layer than produces a vector of the vocabulary size as the output layer. The probability of a word occurring in the context given the target word is calculated by the softmax function on the output layer. The objective function is the maximization of the log probability of a word occurring the context for all word-context word pairs. Similar to (Mnih and Hinton, 2007), the calculation of these probabilities requires a sum over all words of the vocabulary. The hierarchical decomposition proposed in (Mnih and Hinton, 2009; Morin and Bengio, 2005) can be used for speed-up. Instead, Mikolov et al. (2013b) propose the negative sampling similar to (Collobert and Weston, 2008). They use a small number of negative random samples of context-words for target word. They penalize the log probability of negative samples in the objective function. Another improvement of (Mikolov et al., 2013b) over (Mikolov et al., 2013a) is the subsampling of frequent words. Since the most frequent words (e.g. "in", "the", and "a") provide less information than rare words, they discard words depending on their frequency. Experiments show that (Mikolov et al., 2013a) achieve the best results for syntactic and semantic similarity tasks compared to other word vectors and methods proposed in (Mikolov et al., 2013b) accelerated Skip-Gram model improved the quality of the vectors of the rare words.

2.2 SCODE Word Embeddings

(Globerson et al., 2007) introduce the Co-occurrence Data Embedding (CODE) framework that models joint probability of co-occurring discrete random variables using embeddings in a Euclidean space. Maron et al. (2010) extend this

work and introduce the Spherical CODE (SCODE) framework which uses embeddings on a sphere for computational efficiency. Section 3.3 explains the SCODE model in more detail. They apply their model for word-context co-occurrence data which is generated using neighbours of words.

Yatbaz et al. (2012) extend (Maron et al., 2010) by generating co-occurrence data using probable substitutes of words. SCODE word embeddings based on substitue words achieve the best results on unsupervised part-of-speech (POS) tagging.

Baskaya et al. (2013) use a method similar to (Yatbaz et al., 2012) for Word Sense Induction. They separate each occurrence of a word by appending an index. They cluster word embeddings of each word type to induce word senses unlike (Yatbaz et al., 2012) in which word embeddings of all word types are clustered.

(Cirik and Sensoy, 2013) is the first study using SCODE word embeddings in a supervised setup. They induce fine grained word categories using SCODE embeddings. They suggest the use of these fine grained word categories as auxiliary features for dependency parsing. However, the improvements over the baselines were marginal.

Type based studies like (Yatbaz et al., 2012), suffer from ambiguity since they induce one POS category for each word. Cirik (2013) addresses ambiguity in POS induction. Word Tokens of each word type are separated by clustering using their substitute distributions. Sert (2013) proposes a method to solve ambiguity in POS induction. Sert (2013) finds k-nearest neighbours of a word token using substitute word distributions. The distance among substitute word distributions of word tokens determines the nearest neighbours. Using pairs of word tokens and their k-nearest neighbours, a word embedding for each word token is generated by SCODE. This method achieved best token-based POS induction score in English. Yatbaz et al. (2014) introduce another method to solve ambiguity in POS induction. They generate the context embedding of a word token by taking the average of its substitute words embeddings on SCODE sphere. Word token embedding is simply the concatenation of word type embedding and the context embedding of the word token. They achieve comparable results in English. They also achieve

the best published results 12 out of 19 corpora in 15 languages.

Chapter 3

SCODE Word Embeddings with Substitute Word Distributions

In this chapter, I explain the SCODE word embeddings framework proposed in (Yatbaz et al., 2012). In Section 3.1, I explain substitute word distributions that are used to represent the context of a word. In Section 3.2, I explain how substitute word distributions are discretized. In Section 3.3 I introduce Spherical Co-Occurrence Data Embedding framework (Maron et al., 2010).

3.1 Substitute Word Distributions

Substitute word distributions are defined as the probability distributions of observing a word in the context of the target word. They are successful in representing the context of a word (Sert, 2013; Yatbaz et al., 2012, 2014; Yuret and Yatbaz, 2010).

Yuret and Yatbaz (2010) and Yatbaz et al. (2012) define the context of a target word as the sequence of words in the window of size $2n-1$ centered at the position of the target word token. The target word is excluded from the context.

(1) "Steve Martin has already **laid** his claim to that ."

For example, in the sentence (1), the context of the word token '*laid*', for $n = 4$, is ' *Martin has already — his claim to* ' where — specifies the position of the target word token.

Let target word token be in the position 0; the context spans the positions from positions $-n+1$ to $n-1$. The probability of observing each word w in the vocabulary in the context of the target word token is calculated as follows:

$$\begin{aligned} P(w_0 = w|c_{w_0}) &\propto P(w_{-n+1} \dots w_0 \dots w_{n-1}) && (3.1)\\ &= P(w_{-n+1}) \dots P(w_0|w_{-n+1}^{-1}) \dots P(w_{n-1}|w_{-n+1}^{n-2}) && (3.2)\\ &\approx P(w_{-n+1}) \dots P(w_0|w_{-n+1}^{-1}) \dots P(w_{n-1}|w_0^{n-2}) && (3.3)\\ &\propto P(w_0|w_{-n+1}^{-1}) \dots P(w_{n-1}|w_0^{n-2}) && (3.4) \end{aligned}$$

Where, w_i^j is the word sequence from w_i to w_j (for $i < j$) and c_{w_0} is the context of the target word token located at 0 of length $2n-1$.

In Equation 3.1, the right-hand side is proportional to the left-hand side because the context words are fixed. After using the chain rule, Equation 3.2 is obtained from the right-hand side of Equation 3.1. By applying n^{th}-order Markov assumption, only the closest $n-1$ words in each term of the Equation 3.2 are needed which gives the Equation 3.3. The Equation 3.4 is proportional to the Equation 3.3 because the context of the target word is fixed, thus, any term that does not depend on w_0 is fixed. The terms from Equation 3.4 are truncated or dropped near the boundaries of the sentence. (e.g. if 0 is the first word of a sentence, $P(w_0|w_{-n+1}^{-1})$ becomes $P(w_0)$). An n-gram language model provides the probabilities required for Equation 3.4.

Table 3.1 illustrates the substitute distribution of "laid" in (1). I list 5 substitutes from substitute distribution, however, there is a row for each word in the vocabulary in the distribution. Figure 3.1 shows the substitute distribution for sentence (1). Only the top three substitute words are listed.

Table 3.1: Substitute word distribution for "laid" in sentence (1).

Probability	**Substitute Word**
0.191	staked
0.161	established
0.125	made
0.096	proved
0.094	rejected

Word Token	**Substitute**	**Words**				
Steve	Lockheed	0.137	Mr	0.108	Chris	0.106
Martin	Cotterill	0.147	Jones	0.122	Wilson	0.113
has	is	0.215	has	0.172	had	0.161
already	finally	0.192	been	0.138	also	0.097
laid	staked	0.191	established	0.161	made	0.125
his	no	0.550	a	0.112	the	0.081
claim	answer	0.147	response	0.135	claim	0.106
to	to	0.391	on	0.125	against	0.081
that	prophethood	0.169	be	0.154	fame	0.145
.	.	0.671	?	0.143	!	0.121

Figure 3.1: Most probable substitute word types for each context in the sentence (1). The values on the right side are the substitution probability of the word to the left.

3.2 Discretization of Substitute Word Distributions

The Euclidean embedding algorithm I describe in Section 3.3, requires categorical variables co-occurring together as its input. I aim to associate words co-occurring in similar contexts. Although substitute word distributions represent the context of a word, they are continuous vectors. Thus, I transformed substitute distributions into a discrete setting.

I list two solutions proposed to discretize substitute word distributions. The first solution is to use the similarities of substitute word distributions. (Sert, 2013) uses ids of k-nearest neighbors of a substitute word distribution of a word as categorical variables. This method is computationally expensive

Word Token	Substitute Word
Steve	Mr
Steve	Chris
Martin	Coppell
Martin	Wilson
has	had
has	has
already	finally
already	already
laid	made
laid	shown
his	no
his	no
claim	response
claim	testimony
to	to
to	to
that	fame
that	succeed
.	.
.	.

Figure 3.2: Sampling twice from the substitute word distributions of sentence (1).

since it has to calculate distance among substitute word distributions for all word tokens.

The second solution is sampling with replacement word types from substitute word distributions (Yatbaz et al., 2012). It solves problems of the previous approach. However, the number of samples should be chosen carefully, since if the number of the samples are too small, it may fail to capture the characteristics of the distribution.

In Figure 3.2 is an example of a discretization with sampling. Substitute words are sampled from substitute word distributions partly depicted in Figure 3.1[1].

3.3 Spherical Co-Occurrence Data Embedding

This section is a short review of the Symmetric Interaction Model of the Co-occurrence Data Embedding (CODE) (Globerson et al., 2007) and its extension Spherical Co-Occurrence Data Embedding (S-CODE) (Maron et al.,

[1]Note that Figure 3.1 only has 3 most probable substitute distributions while we sampled from the whole distribution in experiments.

2010) frameworks.

I map co-occurrence data generated from the word types and substitute word samples described in Section 3.2 to d dimensional Euclidean space. As an illustration, Figure 3.3 shows an example artificial co-occurrence data and its embedding on a unit-sphere using the S-CODE.

Let X and Y have a joint distribution such that X and Y are categorical variables with finite cardinality $|X|$ and $|Y|$. However we only observe a set of pairs $\{x_i, y_i\}_{i=1}^n$ drawn IID from the joint distribution of X and Y. These pairs are summarized by the empirical distributions $\bar{p}(x, y)$, $\bar{p}(x)$ and $\bar{p}(y)$. Embeddings $\phi(x)$ and $\psi(y)$ can capture the statistical relationship between the variables x and y in terms of square of Euclidean distance $d_{x,y}^2 = \|\phi(x) - \psi(y)\|^2$. Pairs frequently co-occur are embedded closely in d dimensional space because of this statistical relationship.

I used the following extended model Maron et al. (2010) proposed among others in Globerson et al. (2007):

$$p(x, y) = \frac{1}{Z}\bar{p}(x)\bar{p}(y)e^{-d_{x,y}^2} \tag{3.5}$$

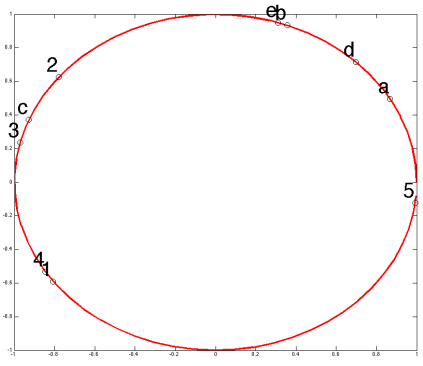
where $Z = \sum_{x,y} \bar{p}(x)\bar{p}(y)e^{-d_{x,y}^2}$ is the normalization term. The log-likelihood of the joint distribution over all embeddings ϕ and ψ can be descibed as the following:

$$\begin{aligned} \ell(\phi, \psi) &= \sum_{x,y} \bar{p}(x, y) \log p(x, y) \\ &= \sum_{x,y} \bar{p}(x, y)(-\log Z + \log \bar{p}(x)\bar{p}(y) - d_{x,y}^2) \\ &= -\log Z + const - \sum_{x,y} \bar{p}(x, y) d_{x,y}^2 \end{aligned} \tag{3.6}$$

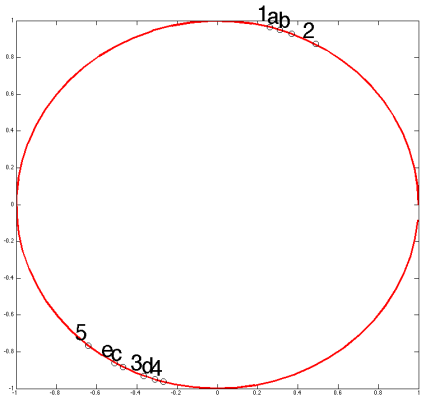
The gradient of the log-likelihood depends on the embeddings $\phi(x)$ and $\psi(y)$, for $x \in X$ and $y \in Y$, and to maximize the log-likelihood, (Maron

X	Y
a	1
a	1
a	2
a	2
b	1
b	2
b	2
b	2
c	3
c	4
c	3
c	3
d	4
d	3
d	4
d	5
e	5
e	3
e	3
e	4

(a)



(b)



(c)

Figure 3.3: (3.3a) An artificial co-occurrence data set. X is a variable over letters from a to e and Y is a variable generates digits from 1 to 5. (3.3b) All embeddings are distributed randomly on the unit circle in the beginning. (3.3c) When the system converges, embeddings of x pairs and y pairs that co-occurred together are much closer to each other compared to the pairs that do not co-occur. For instance a and b co-occurs with 1. After convergence, a and b are close to each other.

et al., 2010) use a gradient-ascent approach. The gradient is :

$$\frac{\partial \ell(\phi,\psi)}{\partial \phi(x)} = \sum_{y} 2\bar{p}(x,y)[\psi(y)-\phi(x)] + \frac{1}{Z}\sum_{y}\bar{p}(x)\bar{p}(y)[\phi(x)-\psi(y)]e^{-d^2_{x,y}} \tag{3.7}$$

$$\frac{\partial \ell(\phi,\psi)}{\partial \psi(y)} = \sum_{x} 2\bar{p}(x,y)[\phi(x)-\psi(y)] + \frac{1}{Z}\sum_{x}\bar{p}(x)\bar{p}(y)[\psi(y)-\phi(x)]e^{-d^2_{x,y}} \tag{3.8}$$

The first sum in (3.7) and (3.8), the gradient of the part with $d^2_{x,y}$ of (3.6) acts as an attraction force between the $\phi(x)$ and $\psi(y)$. The second sum in (3.7) and (3.8) , the gradient of $-\log Z$ in (3.6) acts as a repulsion force between the $\phi(x)$ and $\psi(y)$.

Maron et al. (2010) constrain all embeddings ϕ and ψ lie on the d dimensional unit sphere, hence the name S-CODE. A coarse approximation in which all ϕ and ψ distributed uniformly and independently on the sphere, enables Z to be approximated by a constant value. Thus, it does not require the re-computation of Z after every so many steps.

For the experiments in the work, I use SCODE with sampling based stochastic gradient ascent, a constant approximation of Z and randomly initialized ϕ and ψ vectors.

3.4 Induction Setup

In this Section, I explain induction details of SCODE word embeddingsand how I obtain other word embeddings.

3.4.1 Unlabeled Data

Word embeddings require a large amount of unlabeled data to efficiently capture syntactic and semantic regularities. The source of the data also may have an impact on the success of the downstream task. Thus, I induce word embeddings using large corpora whereas (Sert, 2013; Yatbaz et al., 2012, 2014) use task-specific, smaller corpora.

Following (Turian et al., 2010), I used the RCV1 corpus containing 190M

word tokens (Rose et al., 2002). Although, I followed the same pre-processing technique described in that work, I got 90M word tokens after cleaning.

I induce word embeddings for multilingual experiments explained in Section 4.1.3. I generate embeddings using subsamples of corresponding Tenten Corpora (Jakubíček et al., 2013) for Czech, German, Spanish and Swedish and Wikipedia dump files for Bulgarian, Hungarian. For Turkish, I used a web corpus (Sak et al., 2008). Table 3.2 shows the statistics of unlabeled corpora for the languages used in this study.

Table 3.2: Unlabeled Corpora of Different Languages for Word Embeddings

Language	**Corpus**	**Number Of Words**
Bulgarian	Wikipedia	101M
Czech	Tenten	140M
English	RCV1	90M
German	Tenten	180M
Spanish	Tenten	106M
Swedish	Tenten	113M
Turkish	Web Corpus	180M

3.4.2 Parameter Settings

To generate substitute word distributions, I trained a 4-gram statistical language model (LM) using SRILM (Stolcke, 2002). I used interpolated Kneser-Ney discounting. I replaced words observed less than 2 times with unknown tag. Table 3.3 shows the statistics of language model corpora for each language. I used ukWaC corpora for English (Ferraresi et al., 2008). I used the FASTSUBS algorithm (Yuret, 2012) to generate top 100 substitutes words and their substitute probabilities.

I keep each word with its original capitalization. I sample 100 substitutes per instance. The SCODE dimensions and normalization constant were set to 25 or 50 and 0.166, respectively.

Table 3.3: Unlabeled Corpora for Language Modeling

Language	**Corpus**	**Number Of Words**
Bulgarian	Wikipedia	850M
Czech	Tenten	1.79B
English	ukWac	2B
German	Tenten	1.8B
Spanish	Tenten	2.4B
Swedish	Tenten	113M
Turkish	Web Corpus	1.8B

3.4.3 Other Word Embeddings

I downloaded word embeddings from corresponding studies[2][3][4].(Dhillon et al., 2011; Huang et al., 2012; Turian et al., 2010). I should note that I do not use the context-dependent word embeddings of (Dhillon et al., 2011). These word embeddings are scaled with parameter $\sigma = 0.1$ since Turian et al. (2010) have shown that word embeddings achive their optima at this value. I use 50-dimensional version of each word embeddings in all comparisons.

To induce Skip-Gram NLM embeddings, I ran the code provided on the website[5] of (Mikolov et al., 2010, 2013c) on the RCV1 corpus explained in Section 3.4.1. I changed the words occurring less than 2 times with an unknown tag, thus, I am able to represent words that are not seen in word vector corpus with an unknown word vector. The performance of Skip-Gram NLM and SCODE word embeddings do not change with scaling, thus, I use them without scaling.

[2]http://metaoptimize.com/projects/wordreprs/
[3]http://www.cis.upenn.edu/ ungar/eigenwords/
[4]http://goo.gl/ZXv0Ot
[5]https://code.google.com/p/word2vec/

Chapter 4

Experiments

In this chapter, I detail the experiments. I introduce tasks in which I compared word embeddings, the data used, and parameter choices made.

4.1 Supervised NLP Tasks

In this section, I introduce supervised NLP tasks in which I compare word embeddings. Chunking and NER have become the standard benchmark tasks to compare word embeddings after (Dhillon et al., 2011; Turian et al., 2010). They are described in Section 4.1.1 and Section 4.1.2. In addition, I present Dependency Parsing as my third task which is explained in Section 4.1.3.

4.1.1 Chunking

Chunking is the task of dividing text into syntactically related groups. These groups are non-overlapping, i.e. one word cannot be a part of more than one chunk.

> [The Atlanta-based chemical manufacturer]$_{NP}$ [said]$_{VP}$ [lower prices]$_{NP}$ [hurt]$_{VP}$ [margins]$_{NP}$ [for]$_{PP}$ [most products]$_{NP}$ [.]$_0$

Above is an example sentence divided into chunks. Groups of words between square brackets are examples of chunks. The tag next to the brackets are the types of the chunks. For instance NP stands for noun phrase.

I used CoNLL-2000 Shared task as my benchmark (Tjong Kim Sang and Buchholz, 2000). The data is from the Penn Treebanks which is newswire text from Wall Street Journal (Marcus et al., 1999). The training set contains 8.9K sentences. The development set contains 1K sentences and the test set has 2K sentences.

I use the publicly available implementation of (Turian et al., 2010). It is a CRF based chunker using the features given in Table 4.1. The only hyperparameters of the model was l2-regularization sigma which is optimal at 2. After successfully replicating the results in that work, I ran experiments for new word embeddings.

Table 4.1: Features Used In CRF Chunker. Table is from (Turian et al., 2010). The baseline chunker does not use embedding features.

- Word features: w_i for i in {-2,-1,0,+1,+2}, $w_i \wedge w_{i+1}$ for i in {-1,0}
- Tag features: w_i for i in {-2,-1,0,+1,+2}, $t_i \wedge t_{i+1}$ for i in {-2,-1,0,+1},. $t_i \wedge t_{i+1} \wedge t_{i+2}$ for i in {-2,-1,0}.
- Embedding features: $e_i[d]$ for i in {-2,-1,0,+1,+2}, where d ranges over the dimensions of the embedding e_i.

4.1.2 Named Entity Recognition

The Named Entity Recogniton (NER) task aims to discover named entity phrases containing the names of persons, organizations, and locations.

> "... political talks with [Taiwan]$_{LOC}$, [Foreign Ministry]$_{ORG}$ spokesman [Shenp Guofang]$_{PER}$ told [Reuters]$_{ORG}$..."

Above is an example from NER data. Words inside the square brackets are named entities. Tags next to the brackes are the types of named entitites. For instance, "Foreign Ministry" phrase is an organization name and tagged as ORG.

The data from the CoNLL-2003 shared task (Tjong Kim Sang and De Meulder, 2003) is extracted from RCV1 Corpus. Th training, development, and

test sets contain 14K, 3.3K and 3.5K sentences, respectively. The annotated named entities are location, organization and miscellaneous names. (Tjong Kim Sang and De Meulder, 2003) details the number of named entities and data preprocessing. In addition, (Turian et al., 2010) evaluated word embeddings on an out-of-domain (OOD) data containing 2.4K sentences (Chinchor, 2013).

I used the publicly available implementation of (Turian et al., 2010). It is a regularized averaged perceptron model exploiting features descibed in Table 4.2. After I replicated the results of that work, I ran the same experiments for new word embeddings. It is important to note that, unlike (Turian et al., 2010), I did not use any non-local features or gazetters because I wanted to measure the performance gain of word embeddings alone. The only hyperparameter is the number of epochs for the perceptron. The perceptron stops when there is no improvement for 10 epochs on the development set. The best number of epochs on development set is used for the final model.

Table 4.2: Features Used In Regularized Averaged Perceptron for NER. Table is from (Turian et al., 2010). Word vectors are used the same way as in Table 4.1

.

- Previous two predictions y_{i-1} and y_{i-2}
- Current word x_i
- x_i word type information : all-capitalized, is-capitalized, all-digits, alphanumeric etc.
- Prefixes and suffixes of x_i, if the word contains hyphens, then the tokens between the hyphens
- Tokens in the window $c = (x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2})$
- Capitalization pattern in the window c
- Conjunction of c and y_{i-1}

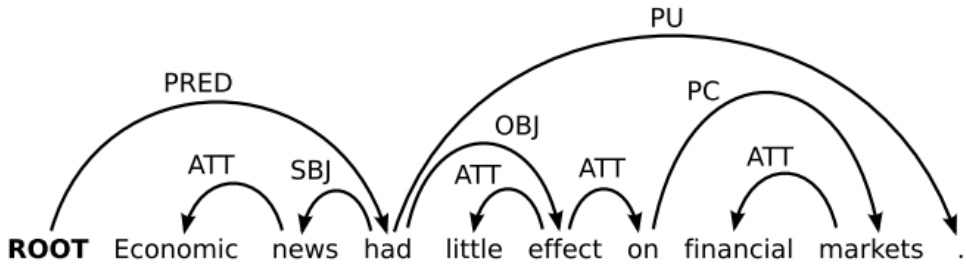


Figure 4.1: An Example Sentence with its Dependency Parse from (Kübler et al., 2009)

4.1.3 Dependency Parsing

Dependency grammar aims to capture syntactic relations in a sentence by defining asymmetric, binary head-dependent relationships among words. These relationships form a dependency tree. Dependency parsing aims to recover the dependency tree given only the sentences and the syntactic features.

Figure 4.1 shows dependency tree of an example sentence. Arrows represent dependency relations from the head word to the dependent word. Arrows also have labels to indicate the type of the dependency relation. For instance, "effect" is dependent of "had" with the subject (SBJ) dependency.

Note that there is an artificial word *root* before the first word in Figure 4.1. Now each word has a syntactic head, as a result, the root node simplifies the formalism of dependency trees and computational implementations.

Recently, Lei et al. (2014) proposed a framework that is capable of incorporating word embeddings in dependency parsing. They reduce the dimensionality of head-modifier feature vectors by learning a tensor of low rank. They are able to combine features from state-of-the-art parsers MST Parser (McDonald et al., 2005) and Turbo Parser (Martins et al., 2013) as well as low-rank tensor features which includes word embeddings. Table 4.3 lists the features. There are two hyperparameters γ and r. The first one balances tensor features and traditional MST/Turbo features. The second one is the rank of the tensor. I set the hyperparameters $\gamma = 0.3$ and $r = 50$ and used standard model to get results comparable result in that work.

Following (Lei et al., 2014), I chose the CoNLL-2008 task (Surdeanu et al., 2008) as my benchmark to compare word embeddings in English. For computational reasons, I fixed the training set to the first 5K sentences of the CONLL 2008 English dataset. However, I conducted experiments using full training set with SCODE embeddings. For multilingual experiments, I chose the CoNLL-2006 Shared Task languages Spanish, Czech, German, Swedish, and Turkish (Buchholz and Marsi, 2006).

Table 4.3: Features Used In Low-Rank Tensor Parser. Table is from (Lei et al., 2014). Word vectors are used the same way as in Table 4.1

.

- Lemmas in the window $L = l_{i-2}, l_{i-1}, l_i, l_{i+1}, l_{i+2}$
- Forms in the window $F = f_{i-2}, f_{i-1}, f_i, f_{i+1}, f_{i+2}$
- POS tags in the window $P = p_{i-2}, p_{i-1}, p_i, p_{i+1}, p_{i+2}$
- Morphologic feature (m_i
- POS bigrams $p_bigram = (p_{i-1}, p_i), (p_i, p_{i+1})$
- POS - lemma pair, (p_i, l_i)
- Morphologic feature - lemma pair, (m_i, l_i)
- POS trigram (p_{i-1}, p_i, p_{i+1})

4.2 Results

In this section, I report the performance of word embeddings. In Section 4.2.1 results of Chunking is presented. In Section 4.2.2 and Section 4.2.3, results of NER and Dependency Parsing are provided, respectively. Before presenting the results, first, I report word token coverage for word embeddings. For each task, an unknown word in the training or test data is replaced with the word embedding of unknown tag. Thus, word embeddings with high coverage suffer less from unknown words, which in turn effects its success. Table 4.4 shows the word token coverage for each task and their correspond-

ing datasets. GCA NLM has the lowest coverage in all tasks, which may explain its level of performance.

Table 4.4: Word token coverage for word embeddings.

	Chunking			NER			Dependency Parsing	
Word Embeddings	Training	Development	Test	Training & Development	Test	OOD	Training	Test
C&W	0.98	0.9832	0.9764	0.9402	0.9359	0.9631	0.9835	0.9856
HLBL	0.9654	0.9675	0.9621	0.9549	0.9503	0.9777	0.9691	0.9674
GCA NLM	0.823	0.8271	0.8139	0.6971	0.6760	0.8208	0.8322	0.8270
LR-MVL	0.9806	0.9839	0.9778	0.9422	0.9380	0.9637	0.9841	0.9862
Skip-Gram	0.9848	0.9877	0.9827	0.9117	0.9075	0.9614	0.9833	0.9852
SCODE	0.9848	0.9877	0.9827	0.9117	0.9075	0.9614	0.9833	0.9852

4.2.1 Chunking

The performance of a chunker is measured with three rates. The first one is precision which is equal to the percentage of detected phrases that are correct. The second one is recall, the percentage of phrases in the data that were found by the chunker. The third one is F1-score which is the harmonic mean of precision and recall.

In Table 4.5, I report the F1-score of word embeddings and the score of the baseline chunker that is not using word embeddings. All word embeddings are 50 dimensional. They all improve the baseline chunker. The best score is achieved by SCODE embeddings. However the difference between SCODE embeddings and LR-MVL is not statistically significant. The difference between SCODE and the baseline is statistically significant ($p < 0.05$). I also conducted experiments to measure the effect of dimensions (ranging 25 to 200) of word embeddings. I observed no significant effect of dimensions in the F1-score.

4.2.2 Named Entity Recognition

Similar to Chunking, the score of a NER system is measured by F1-score. A predicted named entity is correct only if there is an exact match of the corresponding entity in the test file.

Table 4.5: Chunking Results for Word Embeddings. The ones in bold font are the highest scores in their columns.

Word Embeddings	Development Score	Test Score
Baseline	0.9416	0.9379
C&W	**0.9466**	0.9410
HLBL	0.9463	0.9400
GCA NLM	0.9425	0.9402
LR-MVL	0.9458	0.9416
Skip-Gram	0.9400	0.9402
SCODE	0.9430	**0.9429**

Table 4.6 summarizes the results of NER experiments. The first three rows are from (Turian et al., 2010). They report the baseline and the best results for C&W and HLBL embeddings. The baseline system does not use word embeddings as features. The word embeddings in the other experiments are in 50 dimensions. All of the word embeddings significantly improve the baseline system. SCODE embeddings achieve the best score on the Test set and the Out of Domain Test set. However, the difference between the best system and its runner-up is statistically significant only for OOD data($p < 0.05$). Note that RCV1 corpus is the superset of NER training and test data. Thus, C&W, HLBL and SCODE on RCV1 embeddings are from the same data source. I conducted the same experiments for different dimension sizes (ranging from 25 to 200) for SCODE embeddings and did not observe significant changes in scores.

4.2.3 Dependency Parsing

There are a couple of ways to score a dependency parser. The first one is the percentage of exact matching of parsed sentences. The second one is the percentage of words that are connected to the correct head. The third one is F1-measure which evaluates the percentage of dependencies with specific type in the parser output and in the test set in a similar way to NER and Chunking. In this work, I report the second measure, Unlabeled Attachment

Table 4.6: NER Results for Word Embeddings. The ones in bold fonts are the highest scores in their columns.

Word Embeddings	**Development**	**Test**	**OOD**
Baseline	0.9003	0.8439	0.6748
C&W 200-dim	**0.9246**	0.8796	0.7551
HLBL 100-dim	0.9200	0.8813	0.7525
C&W	0.9227	0.8793	0.7574
HLBL	0.9146	0.8705	0.7293
GCA NLM	0.9	0.8467	0.6752
LR-MVL	0.9171	0.8683	0.7323
Skip-Gram	0.9095	0.8647	0.7194
SCODE	0.9207	**0.8835**	**0.7739**

Score (UAS).

Table 4.7 shows the results for word embeddings and the baseline parser which is not using word embeddings. Each word embedding, shows improvements over the baseline parser. However, improvements are not statistically significant, similar to Chunking results. The SCODE embeddings achieve the best scores among others. The difference between SCODE and LR-MVL is not statistically significant. However, the improvement over the baseline is statistically significant for the SCODE embeddings ($p < 0.05$). I conducted the same experiments for different dimension sizes (ranging from 25 to 200) for SCODE embeddings and did not observe significant changes in scores.

I report Multilingual Dependency Parsing scores in Table 4.8. In the first column, the results reported in (Lei et al., 2014) is listed. In the second column, the state-of-the-art results before (Lei et al., 2014). In the third column, the parser using the SCODE embeddings are listed. The SCODE embeddings improve parsers for 5 out of 6 languages and achieve the state of the art results for 4 out of 6 of them.

Table 4.7: Dependency Parsing Results for ConLL 2008 English Data for Word Embeddings. The ones in bold font are the highest scores in their columns.

Word Embeddings	Training Score	Test Score
Baseline	0.9447	0.8976
C&W	0.9332	0.9007
HLBL	**0.9459**	0.9013
GCA NLM	0.9140	0.8985
LR-MVL	0.9308	0.9016
Skip-Gram	0.9397	0.9014
SCODE	0.9444	**0.9028**

Table 4.8: Dependency Parsing Results for ConLL 2006 Languages for SCODE Embeddings. The ones in bold font are the highest scores in their rows.

Language	**Baseline**	**State-of-The-Art**	**SCODE Embeddings**
Czech	**0.9050**	0.9032	0.9038
English	0.9302	0.9322	**0.9344**
German	0.9197	**0.9241**	0.9233
Spanish	0.8800	0.8796	**0.8823**
Swedish	0.9100	0.9162	**0.9165**
Turkish	0.7684	0.7755	**0.7783**

Chapter 5

Conclusion

In this thesis, I analyzed SCODE word embeddings in supervised NLP tasks. SCODE word embeddings were previously used in unsupervised part of speech tagging (Cirik, 2013; Sert, 2013; Yatbaz et al., 2012, 2014) and word sense induction (Baskaya et al., 2013). Their first use in a supervised setting was in dependency parsing (Cirik and Sensoy, 2013), however, the results were inconclusive. (Lei et al., 2014), suffessfully made use of SCODE embeddings in dependency parsing.

I compared SCODE word embeddings with existing word embeddings in Chunking, NER, and Dependency Parsing. For all these benchmarks, I used publicly available implementations. They all were near state-of-the-art solutions in these tasks. The SCODE embeddings improved the baselines significantly. The SCODE embeddings are at least good as other word embeddings or achieved better results.

I analyzed SCODE embeddings in multilingual settings for Dependency Parsing. SCODE embeddings are consistent in improving the baseline systems. Note that other word embeddings are not studied in multilingual settings. SCODE word embeddings and the code studied in this thesis is publicly available.

For future work, I suggest to analyze the richness of syntactic information of word embeddings. Do word embeddings become redundant in supervised settings when other features are available at hand such as POS tags or mor-

phological features? Or is it the other way around, i.e. are word embeddings capture these features in a dense form instead of being complement to these features? We have on-going studies in addressing these issues.

Another future direction is to solve ambiguity in word embeddings. All occurrences of a word is represented with the same vector. However, the syntactic role and meaning of a word changes in context. For instance, *bat* could mean different things depending on the context. Reisinger and Mooney (2010) and Huang et al. (2012) induce context representations for word. Unfortunately, this method computationally intensive. Dhillon et al. (2011) addresses ambiguity by inducing left and right context representation of a word. (Sert, 2013) and (Yatbaz et al., 2014) address ambiguity in unsupervised POS tagging using SCODE framework. The problem with these approaches is that they are either not orthogonal to the method of inducing word embeddings or computationally intensive. Again, we have on-going studies in this direction and preliminary results are promising.

Bibliography

Osman Baskaya, Enis Sert, Volkan Cirik, and Deniz Yuret. Ai-ku: Using substitute vectors and co-occurrence modeling for word sense induction and disambiguation. *Atlanta, Georgia, USA*, page 300, 2013.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

Sabine Buchholz and Erwin Marsi. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164. Association for Computational Linguistics, 2006.

Nancy Chinchor. Muc-7 named entity task definition. 2013.

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

Volkan Cirik. Addressing ambiguity in unsupervised part-of-speech induction with substitute vectors. *ACL 2013*, page 117, 2013.

Volkan Cirik and Hüsnü 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. Collobert and J. Weston. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *International Conference on Machine Learning, ICML*, 2008.

Scott C. Deerwester, Susan T Dumais, and Richard A. Harshman. Indexing by Latent Semantic Analysis. 1990.

Paramveer Dhillon, Dean P Foster, and Lyle H Ungar. Multi-View Learning of Word Embeddings via CCA. In *Advances in Neural Information Processing Systems*, pages 199–207, 2011.

Paramveer S. Dhillon, Jordan Rodu, Dean P. Foster, and Lyle H. Ungar. Two Step CCA: A new spectral method for estimating vector models of words. In *Proceedings of the 29th International Conference on Machine learning*, ICML'12, 2012.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. Introducing and evaluating ukwac, a very large web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54, 2008.

Amir Globerson, Gal Chechik, Fernando Pereira, and Naftali Tishby. Euclidean Embedding of Co-occurrence Data. *J. Mach. Learn. Res.*, 8:2265–2295, December 2007. ISSN 1532-4435.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew 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*, ACL '12, pages 873–882, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Miloš Jakubíček, Adam Kilgarriff, Vojtěch Kovář, Pavel Rychlỳ, and Vít Suchomel. The tenten corpus family. In *International Conference on Corpus Linguistics, Lancaster*, 2013.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127, 2009.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. Low-Rank Tensors for Scoring Dependency Structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2014.

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. Treebank-3. Linguistic Data Consortium, Philadelphia, 1999.

Yariv Maron, Michael Lamar, and Elie Bienenstock. Sphere Embedding: An Application to Part-of-Speech Induction. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1567–1575, 2010.

André FT Martins, Miguel B Almeida, and Noah A Smith. Turning on the turbo: Fast third-order non-projective turbo parsers. *Proc. of ACL*, 2013.

Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98. Association for Computational Linguistics, 2005.

Tomas Mikolov, Martin Karafiát, Lukáš Burget, Jan Cernocky, and Sanjeev Khudanpur. Recurrent Neural Network Based Language Model. *Proceedings of Interspeech*, 2010.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositional-

ity. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013b.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. *Proceedings of NAACL-HLT*, pages 746–751, 2013c.

Andriy Mnih and Geoffrey Hinton. Three New Graphical Models for Statistical Language Modelling. In *Proceedings of the 24th International Conference on Machine learning*, pages 641–648. ACM, 2007.

Andriy Mnih and Geoffrey E Hinton. A Scalable Hierarchical Distributed Language Model. In *Advances in Neural Information Processing Systems*, pages 1081–1088, 2009.

Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *AISTATS*, volume 5, pages 246–252. Citeseer, 2005.

Joseph Reisinger and Raymond 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.

Tony Rose, Mark Stevenson, and Miles Whitehead. The reuters corpus volume 1-from yesterday's news to tomorrow's language resources. In *LREC*, volume 2, pages 827–832, 2002.

H. Sak, T. Güngör, and M. Saraçlar. Turkish language resources: Morphological parser, morphological disambiguator and web corpus. *Advances in natural language processing*, pages 417–427, 2008.

Enis Sert. ord Context and Token Representations from Paradigmatic Relations and Their Application to Part-of-Speech Induction. Master's thesis, Koç University, Turkey, 2013.

Andreas Stolcke. SRILM – An extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286, November 2002.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177. Association for Computational Linguistics, 2008.

Erik F Tjong Kim Sang and Sabine Buchholz. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics, 2000.

Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.

Joseph Turian, Lev Ratinov, and Yoshua 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, 2010.

Peter D Turney and Patrick Pantel. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of artificial intelligence research*, 37 (1):141–188, 2010.

Mehmet Ali Yatbaz, Enis Sert, and Deniz Yuret. Learning Syntactic Categories Using Paradigmatic Representations of Word Context. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 940–951. Association for Computational Linguistics, 2012.

Mehmet Ali Yatbaz, Enis Sert, and Deniz Yuret. Unsupervised instance-based part of speech induction using probable substitutes. In *Proceedings of COLING 2014*. The COLING 2014 Organizing Committee, 2014.

Deniz Yuret. FASTSUBS: An Efficient and Exact Procedure for Finding the Most Likely Lexical Substitutes Based on an N-Gram Language Model. *Signal Processing Letters, IEEE*, 19(11):725–728, Nov 2012. doi: 10.1109/LSP.2012.2215587.

Deniz Yuret and Mehmet Ali Yatbaz. The Noisy Channel Model for Unsupervised Word Sense Disambiguation. *Computational Linguistics*, 36(1): 111–127, 2010.