Advances in Methods and Evaluations for Distributional Semantic Models using Computational Lexicons

By

Meera Satya Hahn

Jinho Choi
Adviser

Department of Mathematics and Computer Science

Jinho Choi

Adviser

Li Xiong

Committee Member

Effrosyni Seitaridou

Committee Member

2016

Advances in Methods and Evaluations for Distributional Semantic Models using Computational Lexicons

By

Meera Satya Hahn

Jinho Choi

Adviser

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Mathematics and Computer Science

2016

Abstract

Advances in Methods and Evaluations for Distributional Semantic Models using Computational Lexicons

By Meera Satya Hahn

Word embedding has drastically changed the field of natural language processing and has become the norm for distributional semantic models. Previous methods for generating word embeddings did not take advantage of the semantic information in sentence structures. In this work we create a new approach to word embedding that leverages structural data from sentences to produce higher quality word embeddings. We also introduce a framework to evaluate word embeddings from any part of speech. We use this framework to assess the quality of word embeddings produced with different semantic contexts and show that sentence structure is rich with semantic information. Our evaluations show that our new word embeddings far out preform the original word embeddings in all parts of speech. Furthermore we examine the task of sentiment analysis in order to demonstrate the superiority of our system's word embeddings.

Advances in Methods and Evaluations for Distributional Semantic Models using Computational Lexicons

By

Meera Satya Hahn

Jinho Choi

Adviser

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Mathematics and Computer Science

2016

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural language processing relies upon the ability to accurately represent words. Recently traditional token representations of words have given way to dense vector representations of words. These word vectors, also called word embeddings, have been a breakthrough innovation in the field of natural language processing (NLP). The purpose of word embeddings is to quantify the semantics of a word in a vector format, that can then be used computationally in an algorithm. The majority of tasks in NLP now use word embeddings as features because of enhanced performance and accuracy. Even larger applications, such as search engines, use word embeddings in their algorithms because it allows them to return better results to users. A shortcoming of current word embeddings is that they have only been robustly evaluated on nouns. While we can assume from the proven success of the embeddings in many tasks that the general architecture of the embedding process has created a sound semantic model overall, we are eager to test the quality of embeddings of all parts of speech.

The small set of verb embedding tests that exist, show that verb embeddings do not carry nearly the same amount of semantic richness that noun embeddings do. Without proper evaluations it is hard to test if this is true for all verbs. Additionally, it creates the question of which other parts of speech are also failing to generate good embeddings. In this work we create new evaluation methods for all parts of speech. We also create higher quality embeddings for all parts of speech by modifying the current word embedding process. By doing this, we will be able to further increase the accuracy of the tasks these distributional semantic models are used in.

Both previous methods of word representation and the current most popular method of word embedding were created based on the distributional hypothesis [23]. The hypothesis states that words in similar contexts have similar meanings. For example, bodies of text that contain words about food such as banana and pasta usually contain these words in the context of the action of eating. Word representations seek to encode this type of semantic similarity in the words vector representation, with the goal that words about food will lie near each other in the vector space. The distributional hypothesis postulates that by examining and comparing the different contexts in which words are used, a model of semantic similarity and difference can be built.

One purpose of this work is to understand and choose how to formulate these different contexts. There is previous work on improving the generation of word embeddings, however most techniques have not experimented with leveraging the structural features of language to choose context words. We examine if the way that the current word embedding model chooses context words is best for certain parts of speech over others. We find that changing the contextual information used in the embedding process, to include structural information, helps to capture more of a word's semantic meaning. The structural components of natural language that we use are dependency relations and predicate structures. We examine how to extract context words from these different structural components. Additionally, we divide up words by parts of speech and examine how the embeddings perform under a variety of contexts.

To further demonstrate the usefulness of word embeddings and to prove the superiority of the structure based embeddings we created, we use the word embeddings in a sentiment analysis task. We create a sentiment analysis system that uses convolutional neural networks. The inputs to the network are the word embeddings of the words in the sentence being classified. We show that using our word embeddings, as input to this network, allow the system to achieve high levels of accuracy.

1.1 Thesis Statement

By exploiting structural sentence information we are able to reach the goal of creating high quality word embeddings. Through creating a novel evaluation framework, for embeddings of all parts of speech, we are able to holistically assess the quality of word embeddings produced over different semantic contexts. Additionally, we prove the usefulness of structure based embeddings through using them in a core natural language processing task.

Chapter 2

Background

There have been multiple methods of creating representations of words. The most commonly used today is Word2Vec and this is the method upon which we base our system. This section starts by defining the different components of natural language we will be using. The section then describes the distributional semantic theory and delves into the architecture of previous systems.

2.1 Natural Language Structures

2.1.1 Parts of Speech

A focus of this work is to study how embeddings from different parts of speech preform and how this differs based on how the context of a word is defined during the embedding process. Examining how parts of speech function in relation to other words in a sentence is crucial to this study. Words are assigned to a part of speech category based on the syntactic function of the word. The main parts of speech studied in this work are listed below with their syntactic function:

- Noun: words to name people, places or things
- Verb: words that name an action or state of being of the subject

- Adjective: words that describe nouns
- Adverb: words that describe adjectives, verbs or adverbs

We examine verbs the closest because verbs are the most important part of speech. A sentence cannot exist without a verb and verbs are often the predicate of a sentence. A predicate is the part of the sentence that describes the actions or state of the subject. Despite verbs being highly significant there are not many verbs; there are under 4,000 verbs used in the English language. For example the WordNet corpus [16] is approximately 76% nouns and only 7% verbs. Since verbs are extremely important lexical items it is very important that they are accurately represented in NLP tasks. Improving the representation of verbs can lead to dramatic improvements of NLP tasks. For this reason we extensively examine verbs in this work.

2.1.2 Dependency Structure

In a sentence there exists a dependency relation for each word. Essentially, each word in a sentence is a lexical item that is linked to other lexical items by a dependency, which places one term as dependent on another. Dependencies can be either syntactic or semantic relations. Semantic relations are harder to identify but are extremely informative. Each word can only depend on one other word but multiple words can depend on the same word. These dependency relations are one type of information that can be extracted from a sentence to form features. The dependency relations within a sentence form an acyclic, fully connected tree where each node is a word of the sentence and the edges are relations. Creating these dependency trees is a core task of Natural Language Processing called dependency parsing.

The task of dependency parsing seeks to identify the dependency relations between words in a single sentence [10]. Dependency parsing is a difficult task in NLP but it has been mostly solved with parsers that reach into the 90th percentile of accuracy. An example of a dependency tree for the sentence “I threw a green ball across the yard to him” is shown in Figure 2.1.

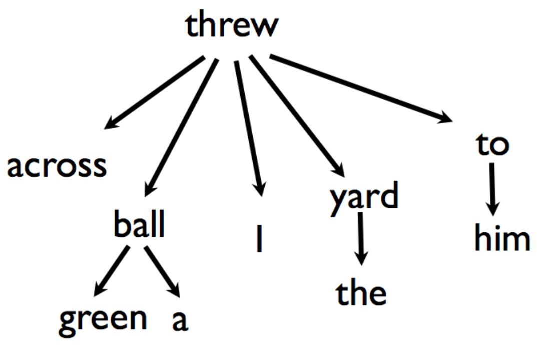


Figure 2.1: The Dependency Tree of Example Sentence
Shows the directed dependency relations between the words in the sentence

Dependency trees are extremely informative because they give information about how words fit together and act in accordance with one another. They also can give information about which words are the most important in the sentence via how many other words are reliant on a particular word. The head of a sentence is arguably the most important word in the sentence while leaf nodes are less important. Additionally, dependency trees give information about which words are most important to a particular word. The graphical distance between two words in a dependency tree can give insight into how important the words are to each other. It is important to note that graphical distance within the dependency tree will often not be the same as topical distance within the sentence. In the above sentence the topical distance between “threw” and “yard” is 6, however the graphical distance is only

1. This shows that it is disadvantageous to exclusively look at topical distance. Another thing to note is that verbs are often the head of the sentence. This further supports the point that verbs are an extremely important part of speech.

2.1.3 Predicate Argument Structure

As stated above, a predicate is the part of the sentence that describes the actions or state of the subject [5]. Predicates are particularly important because they govern the arguments in a sentence. In the sentence “the glacier melted” the word “melted” is the predicate. In the sentence “I threw a green ball across the yard to him” the word “threw” is the predicate. Predicates of a sentence are typically verbs but they can also be other parts of speech. Additionally, predicates are often the head of a dependency tree.

An argument is a participant of the predicate and helps complete the meaning of the predicate. There are different types of relationships between an argument and its predicate. These types are defined by thematic role labels. A list of the different types of thematic role labels and when they are used is listed in Appendix B.

Since there is not a universal set of thematic roles, labels called semantic roles have been created and are defined in respect to verb sense and are given numbers rather than names. The types of semantic roles are listed as follows with a definition of what they represent:

- ARG0: agent
- ARG1: patient, theme
- ARG2: instrument, benefactive, attribute
- ARG3: starting point

- ARG4: ending point
- ARGA: external causer

Predicates in general can be thought of as a function to which there is inputs and outputs. The task of semantic role labeling extracts the variables of this function and creates a predicate structure. These semantic role labels and predicate structure give information about how words in a sentence are related to each other and how they act together to form a cohesive meaning. An example of semantic role labeling on the sentence “I threw a green ball across the yard to him” is shown in Figure 2.2 where “threw” is the predicate of the sentence.

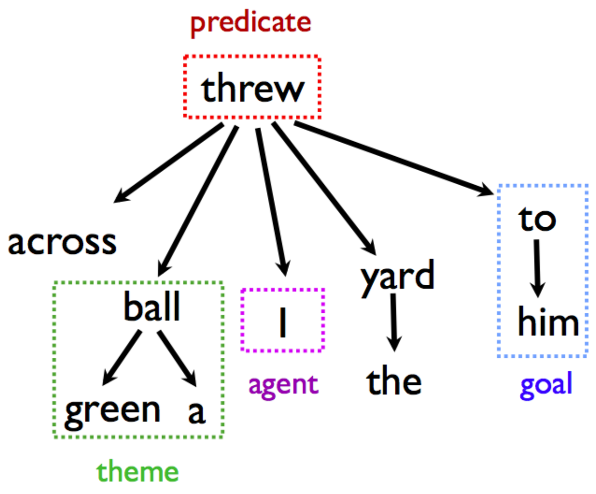


Figure 2.2: Predicate Argument Structure
Shows the dependency parsed tree of the sentence
and highlights the predicate and labeled semantic arguments

2.2 Lexicon Databases

In this work we use two popular lexical databases to create our evaluation framework.

2.2.1 WordNet

The first is called WordNet and is a large and popular English lexical database that contains the majority of words in the English language and covers all parts of speech [16]. The words in this database are grouped into groups called synsets, each of which has a unique concept. The synsets are supposed to capture both lexical and semantic relations. The synsets create a network of words from which it is possible to then measure relations between individual words in the database. Additionally, WordNet distinguishes between senses of a word. Two words with the same form but different senses can be grouped differently within the database.

2.2.2 VerbNet

VerbNet is a database that categorizes all verbs into a set of classes that the database defines. The classes are called verb classes and each verb class is named after the verb that best represents all the verbs in its class. For each verb class, the database defines the thematic roles that can be used for verbs in the class. Additionally, the database defines a set of semantic restrictions on all verbs in the verb class. Essentially semantic restrictions limit the subtypes of roles, within a thematic role, that can be used. The breakdown of information for the verb class Adopt-93, as it is stored in VerbNet, is shown in Table 2.1.

Table 2.1: Attributes of Verb Class Adopt-93

Members	assume	adopt	take_over	take_on	take
Themantic Roles	Agent	Theme			
Semantic Restrictions	animate	organization			

2.3 Distributional Semantics

Distributional semantics is the study and measurement of the semantics of lexical items such as words. Distributional semantic models aim to capture and represent the semantics of these lexical items. Producing these models is a difficult task because the models need to contain the deeper meanings of the lexical items that they are representing. The impact of a rich semantic model is large because it can be used in a variety of other NLP tasks. Humans rely on their understanding of entities, ideas and concepts in the real world to create and analyze text. Distributional semantic models use large corpora of text to generate similar semantic understanding of entities, ideas and concepts that are represented by a singular lexical item such as a word. In this work, we mostly experiment with semantic models of words but also examine how they can be used to form a model for a larger body of text such as a sentence.

Previously in NLP, many tasks were achieved using a distributional semantic model consisting of a tokenized representation of words in sentences. Treating words as tokens is straightforward and robust enough for most simple tasks. Many models will base their labelling off of statistical measures such as prior likelihood and conditional probability. The success of this approach is, to a degree, dependant on the sheer amount of data available to train this simple model. The heavy use of these token based models is partially a result of the observation that simple models trained on enormous datasets generally outperform more complicated models trained on less data.

Token representation, however, does not contain much semantic information about the actual language. Each token is simply an index in a corpus. All relationships between words, besides location relations, are almost entirely ignored in this kind of analysis. Using this representation we might be able to discover that “rose” is a noun, but we could not find out that it is very similar to the word “daffodil”. This type of information is important in more complicated tasks that rely on the sentiments of language, such as question answering and sentiment analysis.

Recently the traditional token representations of words have given way to a new distributional semantic model consisting of dense vector representations of words. These word vectors are also called word embeddings. The shortcoming of current models is that they only have generated high quality vectors for nouns but not for other parts of speech. In this work we seek to improve the quality of vectors for other parts of speech and analyze the shortcomings of current approaches on different parts of speech.

The premise of distributional semantics is that valuable information can be found in the way that words co-occur. The hypothesis from Harris [23] states that words that are used in similar contexts have similar semantic and functional meaning. Therefore the way in which a particular model defines the context of a word highly impacts the way the word is processed. It is very relevant then to define what type of context a model is looking at.

2.4 Related Work

2.4.1 Skip Gram and Continuous Bag of Words

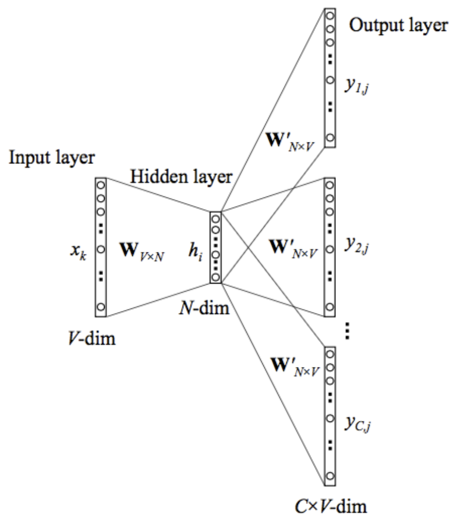
The most heavily used model of distributional semantics, creates dense vectors called word embeddings. Word embedding is a process of mapping words into high dimensional vectors. This mapping allows for reasoning and manipulation of words. This model performs very well in a variety of tasks and, perhaps more interestingly, allows for logical operations through vector manipulation and arithmetic. As Mikolov et al. observed, if the vectorized representation of the word “king” subtracts the vector for “man” and then adds the vector for “woman”, the vector most similar to the resulting vector is for the word “queen”.

In other words: “king” - “man” + “woman” = “queen”.

The current state-of-the-art word embeddings are created by Mikolov’s skip-gram architecture using a negative sampling model. This model provides a very efficient model for generating high-quality embeddings, and is implemented in the Word2Vec package [21].

The skip-gram and continuous-bag-of-words (CBOW) models are built around a neural network with a single hidden layer. For both of these models the context is defined by the preceding and succeeding words around the original word. CBOW takes a context and tries to predict the token that belongs to it. For example, input may be w(i-2), w(i-1), w(i+1) and w(i+2) where the desired output will be w(i) this is shown in Figure 2.4. On the other hand, skip gram takes a single token w(i) and predicts the context w(i-n), ... , w(i-1), w(i+1), ... , w(i+n) this is shown in Figure 2.3. In simple terms skip gram is predicting the context of a word given the word itself and CBOW is predicting a word given the context of the word [21]. In the continuous-bag-of-words model “the order of the words in the history does not influence the projected [vector]” [21]. This discount of word order as a feature in addition to lack of dependency structure can be attributed to problems with finding semantically meaningful representations of verbs.

For each prediction during training, the skip-gram architecture has to update every vocab element in a very large vector. This is a computationally exhaustive procedure. To combat this problem, Mikolov uses a process called negative sampling. Negative sampling



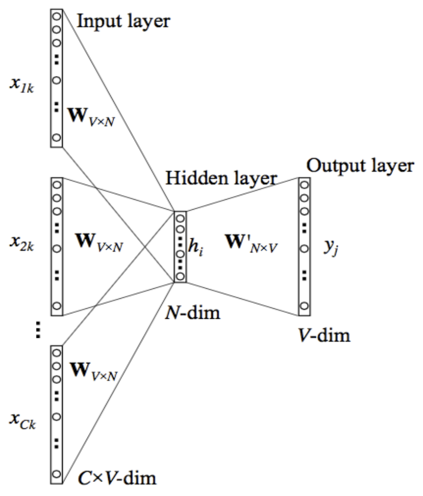


Figure 2.3: (on left) Skip-Gram

Figure 2.4: (on right) Continuous Bag Of Words

then randomly samples only a few of the incorrect possibilities for context words from the entire vocab and forces only the output of these possibilities to zero. The negative sampling approach only applies sampling to incorrect possibilities. For all of the correct context words the architecture forces the probability of the label to one.[20]

2.4.2 Dependency Based Word Embeddings

The skip-gram implementation by [20] defines context as the tokens before and after a given word. The general architecture of skip-gram, which involves predicting the context of a word given the word itself, can be abstracted to use different definitions of what constitutes the context of a word. Levy and Goldberg take this approach by substituting the bag-of-words context for a dependency based context. They define the dependency based context for a word, W, as the head, H, of W and the syntactic dependency label. They call the syntactic dependency label a modifier (e.g. nsubj and dobj). For any relation containing a preposition, the head is then automatically connected to the object of the preposition by a dependency arc, which has the preposition as the dependency label.

An advantage of this approach over a bag of words based skip gram (BOW-SG) model is that it includes words in the context that would otherwise be too far from the word to be included. Take the sentence “Jerry Smith drove home with his car.” A window is used to determine what tokens compose the vector for each word. If a window of size 2 is used, the vector for the word drove is built with the information from tokens “Jerry,” “Smith,” “home,” and “with.” The BOW-SG model would fail to include the important and meaningful relationship between “drove” and “car.” In this same example, a dependency structure would link “drove” to “with” and “with” to “car,” thus including this structure would build more information into the word embeddings.

A shortcoming of this approach [14] is that the context for a word includes only the head of that word W and the dependents of the word W. The paper does not test approaches using additional structural information such as dependency tree siblings or children of the word W. The paper, in which Levy and Goldberg describe their approach for dependency based contexts, does not show any experimentation with including varying amounts of dependency structure information [14].

Additionally, Levy and Goldberg center their evaluations on top of analogy tests that contain almost only nouns. Since Word2Vec [20] already produced semantically relevant word embeddings, the significance of Levy and Goldbergs work was to show that dependency based noun embeddings differ in terms of functional and topical similarity from topographical based noun embeddings. The paper specifically delves into the differences between measures of similarity (functional) and relatedness (topical). The findings of the experiments show that dependency based contexts place words in a vector space where words are closest to functionally similar words.

The paper fails to look heavily into verbs, which is unfortunate since the semantics of verbs are highly based on functionality. In this work we seek to show that different types of dependency based and structural based contexts can be used during the word embedding process to capture semantic functionality and that this constitutes a better word embedding.

Chapter 3

Approach

Our word embedding system is based off the Word2Vec [21] architecture. To implement our distributional semantic model, we modified the vector space model of EmoryNLP in nlp4j [9]. The vector space model in EmoryNLP is a Java implementation of both the Skip Gram and Continuous Bag of Words architectures of Word2Vec.

3.1 System Overview

We first ran through the entire corpus and did tokenization, dependency parsing, semantic role labeling, and part of speech tagging. The corpus that we used included the entire New York Times corpus, which contains over 1.8 million articles that were written between 1987 and 2007. The corpus also included the Wikipedia data dump, which contains all the Wikipedia entries and over three billion words. Word embedding requires a lot of training data to capture the full depth and generalizability of each word. It is also important to have corpus because it allows the system to create a model with a larger vocabulary. Having a large model vocab is important because in tasks such as sentiment analysis, which is performed later in this work, when a word used in the task does not have a vector in the model, a vector is randomly initialized. The Wikipedia dump and the New York Times gave us enough training data to produce meaningful embeddings. The corpus gives us a total of 6,406,418,032 words and 1,531,163 unique word types.

After doing the initial cleaning and parsing of the data, we used the Emory NLP API of readers and objects to read and store all the dependency, Part of Speech (POS) and Semantic Role Label (SRL) information for each word in the corpus. In all experiments the reader used 12 threads.

In our experiments we focused only on the Skip Gram architecture. During the creation of the word embeddings, the algorithm traverses each word W in each sentence of the training data. For each word, its corresponding vector was fed into the one layer neural network. The output of the network is C number of vectors, where C is the number of context words. Each one of the output vectors corresponds to one of the context words. Essentially, the network predicts all context words for the word W.

By changing the context window we expected to see changes in how words were distributed through the resulting vector space. We hoped that certain contexts would do a better job at creating word embeddings that mapped words, that are considered semantically similar in English, close together. We ran a series of experiments where we changed how the algorithm picked the context words. We ran the following experiments where the context window was composed of:

1. **DEP1**: the first order dependencies of W

2. **DEP1H**: the first order dependencies of W and the dependency head of W

3. **DEP12**: the first and second order dependencies of W

4. **DEP12H**: the first and second order dependencies of W and the dependency head of W

5. **DEP1SIB1**: the first order dependencies of W, the rightmost sibling of W and the leftmost sibling of W

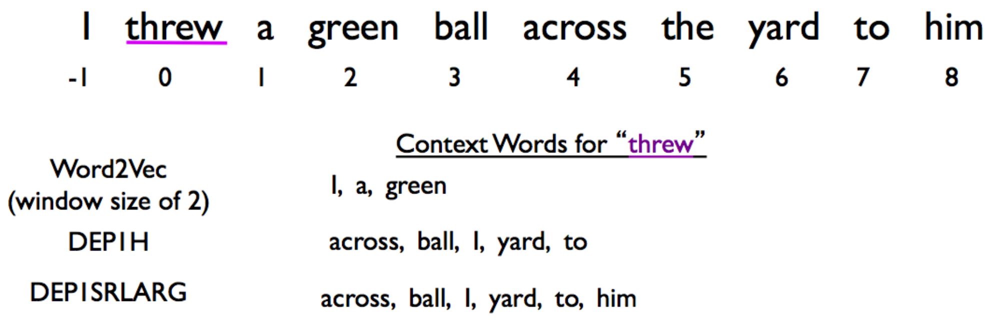
6. **DEP1ALLSIB**: the first order dependencies of W, all siblings of W

7. **DEP1SRLH**: the first order dependencies of W, the semantic head of W

8. **DEP1SRLARG**: the first order dependencies of W, all the semantic arguments of W

To make it easier to cite each of these experiments we have given a shortened name to each of these experiments which is listed next to the experiment description.

For example, let us use the sentence "I threw a green ball across the yard to him" The dependency parsed version of the sentence is shown in Figure 2.1. To create the word embedding for the word "threw" using the skip gram architecture and various experiments is shown in Figure 3.1. Additionally, Figure 3.2 shows a diagram of the system architecture in the DEP1H experiment.



I	threw	a	green	ball	across	the	yard	to	him
-1	0	1	2	3	4	5	6	7	8

	Context Words for "threw"
Word2Vec (window size of 2)	I, a, green
DEP1H	across, ball, I, yard, to
DEP1SRLARG	across, ball, I, yard, to, him

Figure 3.1: Example of Selection of Context Words
Shows the context words that would be selected
for the word "threw" given the three listed experiments

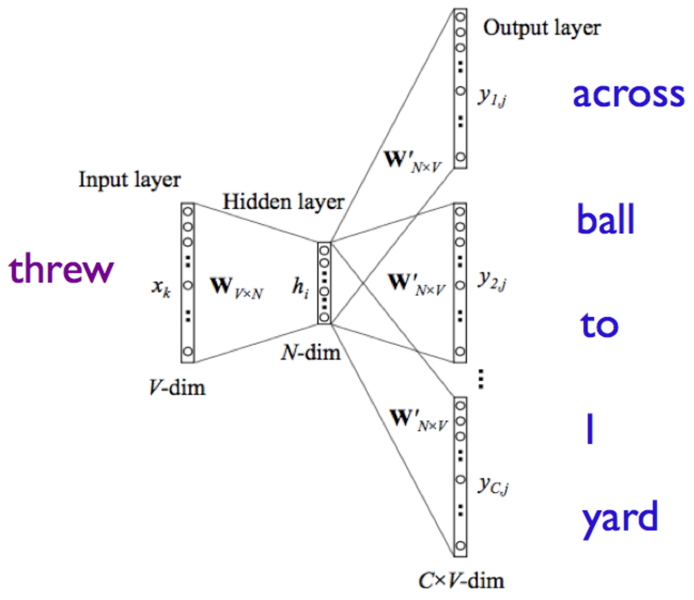


Figure 3.2: Diagram of Skip-Gram based System Architecture for the DEP1H experiment

It is important to note that in some of these experiments, some words may have few or no context words. For example, for experiment DEP1, if a word is a leaf node in a dependency tree then there will be no 1st order dependencies and thus have no context words. In section 4.1.5 we explore the size of the context window for all parts of speech during all experiments and examine how this affects performance of the produced vectors.

We chose the context experiments by specifically thinking of how sentences can be mapped to tree structures. We hoped that using the tree data would produce more semantic information than looking at the sentence linearly, as the original Word2Vec algorithm does.

For all our experiments we held the following variables constant. Our algorithm discards all words that appear less than 5 times in the corpus. Our algorithm runs for a maximum of 5 training iterations negative subsample size of 5. The initial learning rate, alpha, is set to .025. At each training iteration, alpha was adjusted by the following formula that we defined in our algorithm.

$$alpha = .025 * \max(.0001, 1 - \frac{W_g}{i * W_t + 1}) \tag{3.1}$$

where Wg is the number of words traversed so far over all threads, Wt is the number of words in the vocab and i is the number of iterations gone through.

Chapter 4

Experiments

We used multiple lexical databases and methodologies to test the strength of the word embeddings from our various experiments. We used multiple methods to evaluate the strength of embeddings within one experiment and analyzed sources within the lexical databases to give insight into why certain words consistently produced better embeddings than others. The section both goes over the methodology of our evaluations and results of our evaluations. Additionally, the section critically analyzes and draws conclusions from the results.

4.1 WordNet Evaluations

4.1.1 WordNet Similarity

WordNet is a large and popular English lexical database that contains the majority of words in the English language and covers all parts of speech [16]. The words in this database are grouped into synsets sets, each of which has a unique concept. The synsets are supposed to capture both lexical and semantic relations. The synsets create a network of words from which it is possible to then measure relations between individual words in the database. Additionally WordNet distinguishes between senses of a word and the same word form with different senses can be grouped differently within the database.

There are multiple different algorithms to measure the semantic similarity between a pair of words in WordNet. Since WordNet can be represented as a graph, these approaches treat the words as nodes of the network. In this work we limit our evaluations to three popular measures of word semantic similarity based on WordNet. We use the LCH, WUP, and LIN methodologies. In our experiments we utilized the WS4j API which contains implementations of these three measurement methodologies [7].

LCH was developed by Leacock and Chodorow [3]. The measurement of similarity between two words is based on the length of the shortest path between the two synsets containing the words. This method limits attention to IS-A links and "scales the path length by the overall depth D of the taxonomy" [7]. WUP was developed by Wu and Palmer and it calculates the measurement of relatedness based on the depth of the two synsets and the depth of the LCS [24]. LIN was developed by Lin and it is based on another methodology by Jiang and Conrath who use the notion of information content [4] and the equation is as follows:

$$\frac{(2)LC_{LCS}}{IC_{synset1} + IC_{synset2}} \tag{4.1}$$

"where IC(x) is the information content of x. One can observe, then, that the relatedness value will be greater-than or equal-to zero and less-than or equal-to one" [7].

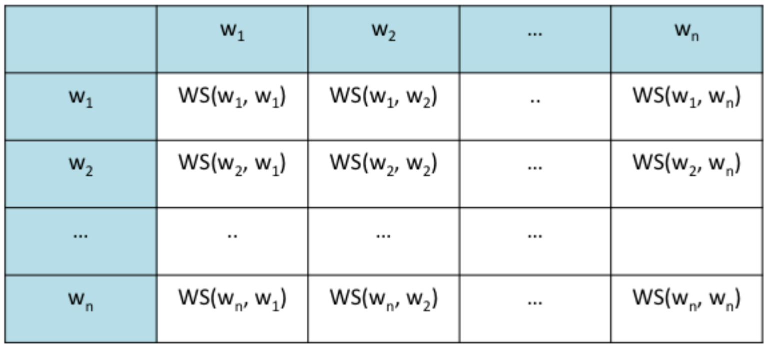
4.1.2 Evaluation Methodology

We created lists for each of the most common words for adjectives, adverbs, nouns, and verbs. To obtain these lists we traversed the entirety of the corpus and looked at the POS tag of each word. We created a map for each POS. The map held words of that POS, mapped to the number of times they were seen in the corpus. We filled out these maps while traversing the corpus. After filling out these maps, we sorted each map in order of the number of times a word was seen. After sorting we saved only the top 5000 most seen words for each POS. When manually looking through these lists we found that some data cleaning was necessary. We did the cleaning by checking that each word in each list also existed in WordNet and removing those that did not. After doing this there was no need for further data cleaning. The final word counts for each POS list are as follows.

1. Adjectives: 4682
2. Adverbs: 4665
3. Nouns: 4827
4. Verbs: 2904
5. Combined Pos: 3966

For each POS list, we used each of the three WordNet similarity measurements to create confusion matrices, where the values of the matrix at a given index corresponded to the WordNet similarity between the word of that column and the word of that row. We then normalized the values of the matrix. This process resulted in a similarity matrix for each POS list and for each WordNet similarity measurement. An example of what one of these matrices would look like is shown in Figure 4.1, where WS(x,y) is the word similarity between x and y. Also note that in this figure WS(x,y) produces the same value as WS(y,x).

Then we took the embeddings produced by each context structure experiment, and used them to make a similarity confusion matrix for each POS list. The values of the matrices at a



	w_1	w_2	...	w_n
w_1	WS(w_1, w_1)	WS(w_1, w_2)	..	WS(w_1, w_n)
w_2	WS(w_2, w_1)	WS(w_2, w_2)	...	WS(w_2, w_n)
...	..	...	...	
w_n	WS(w_n, w_1)	WS(w_n, w_2)	...	WS(w_n, w_n)

Figure 4.1: WordNet Similarity Matrix Example

given index corresponded to the cosine similarity between the word vector of that column and the word vector of that row. We then normalized the values within each matrix. This process resulted in a similarity matrix for each POS list and for each context structure experiment.

We then wanted to compare the word similarity matrices for the context structure experiment (CSE) and the WordNet similarity (WS) matrices. There are multiple avenues of comparison and we originally tried doing a sum squared error between each CSE matrix and each WS matrix, where we used the WS matrix as the correct matrix, with respect to which, we calculated the error. We decided that using sum squared error was not a good comparison for the matrices because the measurements of cosine similarity and WordNet similarity were too different. Ranking correlation was a much more informative and significant comparison for the matrices. Rank correlation is the measurement between rankings of the same variable, where a ranking is the assignment of the labels, 1st, 2nd, nth, to each observation of the same variable. The rank correlation coefficient is a measurement of the similarity between two sets of rankings. We created a similarity ranking for each word, where all the other words were ranked in terms of similarity to that word. We created these similarity rankings for each word in each similarity matrix. We then used the rank correlation coefficient across the rankings of the matrices we are comparing.

4.1.3 Ranking Correlation

We measured ranking correlation between two matrices by going through the matrices one row at a time. In the matrices a row represents a single word, with all other words in the vocab ranked in order of similarity to this word. We took the calculated similarity score and sorted the words from most similar word to least similar word. We then looked at the ranking order of the words for both matrices and performed the ranking correlation metric. Since we could only do ranking correlation for one row of the matrix at a time, we computed the average of the ranking correlations for each row and used that as the total ranking correlation of the matrix. Ranking correlation for an individual word lies between -1 and 1, with 1 being the best possible score and identical ranks. A ranking correlation can be negative when two lists of ranks are inverses of each other.

We used two popular rank correlation methods called Kendalls Rank Correlation and Spearmans Rank Correlation. Spearmans Rank Correlation does not make any assumptions about the distribution of the data and is best used on ordinal data. Spearmans is similar to the Pearson Correlation but Pearson is best used on interval data. Spearmans is a nonparametric measure of the degree of association between variables. The following formula is used to compute Spearmans coefficient:

$$p = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{4.2}$$

where p is the ranking correlation coefficient, di is the difference between the ranks of corresponding values Xi and Yi and n is the number of values in each data set, in other words the length of each ranking vector.

Kendalls rank correlation is a nonparametric measure of the degree of dependence between two variables. The following formula is used to compute Kendalls coefficient:

$$p = \frac{n_c - n_d}{\frac{1}{2}n(n-1)} \tag{4.3}$$

where p is Kendalls coefficient, nc is the number of concordant (ordered the same) pairs, nd is number of discordant (ordered differently) pairs, and n is the number of values in each data set.

The results of the ranking correlations between all matrices are shown in Table 4.1 and Table 4.2.

	EmoryW2V	**DEP1**	**DEP1H**	**DEP12**	**DEP12H**	**DEP1SIB1**	**DEP1ALLSIB**	**DEP1SRLH**	**DEP1SRLARG**
noun									
WUP	0.002334838	0.010145907	0.012681005	0.006229027	0.007760952	0.005079918	0.016878502	0.009249868	0.007892326
LIN	0.006554193	0.025176051	0.029649062	0.012859081	0.016696788	0.009607266	0.037541368	0.022278363	0.01993255
LCH	0.001532491	0.008705386	0.011343738	0.005387621	0.006800848	0.004102623	0.015702324	0.007809994	0.006755566
adjective									
WUP	-0.046465126	-0.02293298	-0.01940994	-0.029841536	-0.028420201	-0.025816303	-0.009760687	-0.024603745	-0.028982142
LIN	-0.046098203	-0.023445878	-0.01978931	-0.030520022	-0.028762131	-0.027061267	-0.010249259	-0.025255805	-0.029877578
LCH	-0.046383838	-0.022820065	-0.019406427	-0.029916208	-0.028306369	-0.025751875	-0.009716675	-0.024526921	-0.028958502
adverb									
WUP	4.41685E-05	-0.004179959	0.023550489	9.3957E-05	0.001904147	-0.008183823	0.008146122	-0.004590632	-0.005176605
LIN	6.72539E-05	-0.013162973	0.010427748	-0.009443542	-0.007668107	-0.017257574	-0.000512881	-0.013634445	-0.014237758
LCH	6.81363E-05	-0.004345601	0.023456104	-1.86201E-05	0.001856518	-0.00824885	0.008029164	-0.004658949	-0.00528443
verbs									
WUP	0.019678145	0.051107308	0.040254008	0.037253208	0.038379081	0.044502025	0.050229722	0.050152014	0.04937911
LIN	0.029770669	0.141635079	0.102302894	0.095378533	0.098343152	0.116092163	0.136537524	0.137088719	0.135258667
LCH	0.010619012	0.060383828	0.040477652	0.039515171	0.041584723	0.048461369	0.06048967	0.058247727	0.056743657
average									
WUP	-0.024407975	0.034140275	0.057075562	0.013734655	0.019623979	0.015581817	0.065493658	0.030207505	0.023112689
LIN	-0.009706087	0.130202279	0.122590394	0.06827405	0.078609701	0.081380587	0.163316752	0.120476833	0.111075881
LCH	-0.034164199	0.041923548	0.055871067	0.014967964	0.021935719	0.018563268	0.074504483	0.036871851	0.02925629

Table 4.1: WordNet Kendalls Rank Correlation Evaluation

4.1.4 Result Analysis

Looking at Table 4.1 and Table 4.2, we can examine which types of embeddings did the best for each part of speech. One immediate observation we can make is over the differences between using Kendalls and Spearmans ranking correlation. We can clearly see that the two ranking correlations give different numerical results for all the matrix comparisons. We can also observe that despite this, the two ranking correlations for a given POS, both result in

	EmoryW2V	**DEP1**	**DEP1H**	**DEP12**	**DEP12H**	**DEP1SIB1**	**DEP1ALLSIB**	**DEP1SRLH**	**DEP1SRLARG**
noun									
WUP	0.003500179	0.015202139	0.018994869	0.009336212	0.011630819	0.007593662	0.025264565	0.014073054	0.011828739
LIN	0.009819439	0.037716979	0.044371939	0.019286759	0.025028944	0.014357013	0.056090768	0.033412487	0.029887544
LCH	0.002296253	0.013039684	0.016992504	0.008076199	0.010191908	0.006128364	0.023505933	0.011708138	0.010126313
adjective									
WUP	-0.067973988	-0.034368459	-0.028998413	-0.044726111	-0.042579685	-0.038717989	-0.014694022	-0.036852231	-0.043374756
LIN	-0.067428383	-0.035132217	-0.029565836	-0.045744568	-0.0430913	-0.040587587	-0.015433643	-0.037823345	-0.044704883
LCH	-0.067852962	-0.034199658	-0.028991893	-0.044837713	-0.0424098	-0.038619187	-0.014624742	-0.036736509	-0.043338843
adverb									
WUP	6.61286E-05	0.000348218	0.040333253	0.001958177	0.004626927	-0.011431714	0.012864709	-0.000295248	-0.001492765
LIN	0.000101503	-0.012987871	0.020949891	-0.012231952	-0.009625922	-0.024946393	-1.81205E-05	-0.013722069	-0.014950668
LCH	0.00010265	0.000101119	0.040190271	0.001787143	0.004555178	-0.011532233	0.012689293	-0.000400886	-0.001653277
verbs									
WUP	0.029523469	0.076321812	0.06017965	0.055704124	0.057367533	0.06648196	0.07494362	0.074898687	0.073755821
LIN	0.044955052	0.208826391	0.151376644	0.141371602	0.145571268	0.17145278	0.200471304	0.202337597	0.199745552
LCH	0.015980665	0.090133676	0.060479238	0.059060928	0.062129506	0.072327784	0.090148846	0.086955049	0.084731097
average									
WUP	-0.034848836	0.05750371	0.09050936	0.022272402	0.031045593	0.02392592	0.098378872	0.051824262	0.040717038
LIN	-0.01255239	0.198423282	0.187132638	0.102681841	0.117882989	0.120275813	0.241110309	0.184204669	0.169977545
LCH	-0.049473393	0.069074821	0.08867012	0.024086557	0.034466792	0.028304727	0.11171933	0.061525792	0.04986529

Table 4.2: WordNet Spearmans Rank Correlation Evaluation

the same context embedding matrix having the best ranking correlation against the WordNet matrices. Additionally, we can see that the type of context embedding matrix that gets the best ranking correlation score for a given POS, is constant across the different WordNet matrices. Since using different metrics yields the same results, we can be extremely confident in our conclusions of which experiments created the best embeddings for each POS. We can conclude that for all parts of speech using a topical contextual window, as the original Word2Vec [21] architecture does, never produces the best embeddings.

Our evaluations of WordNet comparison clearly show that using different context words creates better embeddings for certain POS. This shows that different parts of speech rely on different semantic information. These results can be attributed to the fact that certain parts of speech have less first order dependents than others and that certain parts of speech do not typically have semantic role arguments. We explore the size of the context window for different parts of speech and speculate how this affects the produced embedding in section 4.1.5.

Verb embeddings performed best with the first order dependents as the context words, experiment DEP1. Increasing the context window to include the second order dependents decreased the success of the embeddings. This is probably due to having too many context words and therefore too much information. In future work a direction to explore is running an experiment that includes both the first and second order dependents in the context window, but weights the first order dependents with a higher weight than the second order dependents. While the verb embeddings performed the best with the first order dependents, the DEPALLSIB, DEP1SRLH, and DEP1SRLARG experiments also got good results.

The experiments that placed the semantic role arguments in the context windows only performed well for verbs. This was expected because usually only verbs are predicates and thus only they will have semantic role arguments. This means that verbs are the only part of speech gaining more information from these types context windows. Since words of multiple parts of speech are being trained together, having less informative word vectors for certain parts of speech will also affect verbs even if verbs are able to pick good context words.

The experiments that placed the semantic heads in the context windows performed better overall than the experiments that placed the semantic role arguments in the context windows. We think this is because there are more words that have a semantic head than a set of semantic arguments. This is logical since usually only verbs can have semantic arguments whereas all parts of speech can have semantic heads. This creates better embeddings overall because having more informative vectors for all parts of speech increases individual parts of speech vectors.

4.1.5 Context Window Size Evaluation

To help interpret the results, we decided to go through the entire corpus and get the average context window for each part of speech. We found the following results in Table 4.3. We can confirm that none of the adjectives, adverbs and nouns in our corpus have semantic role arguments. We know this because for experiment DEP1SRLARG, for those parts of speech, the size of the context window is the same as in the DEP1 experiment. Thus only verbs have an advantageous context window in the DEP1SRLARG experiment.

POS	**Adj.**	**Adverb**	**Noun**	**Verb**
DEP1	0.28507	0.12623	1.35436	3.00587
DEP1H	1.28507	1.12623	2.29998	4.00587
DEP12	0.44210	0.19603	1.69356	4.12976
DEP12H	1.44210	1.19603	2.69356	5.12976
DEP1SIB1	1.49963	1.42452	1.60495	3.94874
DEP1ALLSIB	2.12562	3.10883	2.18363	5.28594
DEP1SRLARG	0.28507	0.12623	1.35436	3.00587

Table 4.3: Average Context Window Sizes

4.2 VerbNet Evaluations

4.2.1 Evaluation Methodology

One of our evaluations utilized an English database called VerbNet [11]. VerbNet is a verb lexicon that organizes verbs into hierarchical classes and clusters. For each defined verb class in VerbNet, the thematic roles that correspond to semantic role labels are listed as well as syntactic and semantic frames.

The classes in VerbNet form a baseline of how to group verbs. In our experiments we utilized the JVerbNet API which has implementations in Java to easily access the VerbNet Database [11]. Using the VerbNet classes we wanted to see if certain classes on average contained verbs for which our word embedding algorithm was creating better vectors. To do this we went through each verb class and got the average ranking correlation of each verb in that class. Then we sorted the verb classes by average ranking correlation. We used the LIN [4] correlation metric because in our WordNet evaluation method the LIN correlation metric gave the best results. We additionally ran the same evaluation with WUP [24] and LCH [3] correlation metrics and found extremely similar results so the tables for those are not included.

There were a total of 265 verb classes tested. We did this evaluation for both the EmoryNLP Word2Vec embeddings and the DEP1 structure embeddings. We choose to look at the DEP1 embeddings over the embeddings produced by the other experiments because, from the WordNet evaluations, we found that DEP1 embeddings were the best for verbs. The bottom 10 and top 10 classes for both DEP1 and EmoryNLP Word2Vec are listed in order with their average ranking correlation in Table 4.5 and Table 4.4.

Bottom Classes Name	**Rank**	**Top Classes Name**	**Rank**
267 light_emission-43.1	-0.0151	522 succeed-74	0.1311
268 stalk-35.3	-0.0077	523 relate-86.2	0.1335
269 appeal-31.4	-0.0075	524 dedicate-79	0.1395
270 ferret-35.6	-0.0073	525 learn-14	0.1401
271 animal_sounds-38	-0.0064	526 exist-47.1	0.1402
272 wink-40.3.1	-0.0029	527 continue-55.3	0.1498
273 captain-29.8	-0.0028	528 declare-29.4	0.1503
274 entity_specific_modes_being-47.2	-0.0027	529 empathize-88.2	0.1563
275 bulge-47.5.3	-0.0026	530 involve-107	0.1573
276 weather-57	-0.0021	531 consider-29.9	0.1642

Table 4.4: Verb Rankings DEP1

The tables show that certain verb classes that are in the top and bottom of average ranking correlations of DEP1 also appear in the top and bottom of average ranking correlations of

Bottom Classes Name	**Rank**	**Top Classes Name**	**Rank**
exhale-40.1.3	-0.1473	exist-47.1	0.2397
waltz-51.5	-0.1312	continue-55.3	0.2406
ferret-35.6	-0.0988	dedicate-79	0.2407
funnel-9.3	-0.0936	succeed-74	0.2464
wink-40.3.1	-0.0926	accept-77	0.255
change_bodily_state-40.8.4	-0.0911	involve-107	0.2742
spank-18.3	-0.0858	adopt-93	0.2776
curtsey-40.3.3	-0.0819	learn-14	0.282
snooze-40.4	-0.0815	relate-86.2	0.2947
drive-11.5	-0.0805	cooperate-73	0.299

Table 4.5: Verb Rankings EmoryNLP Word2Vec

Emory Word2Vec. This made us curious to see if there was anything similar between the verb classes that consistently had better or worse embeddings across different context windows. Since VerbNet gives information on thematic roles of each verb class we thought we would first investigate to see if there were any patterns of thematic roles. Thematic roles are a set of categories that groups verbs in terms of the predicate structures of the verbs. A list of the types of thematic roles is in Appendix B. Verbs can have multiple thematic roles.

We used the TF-IDF measure on thematic roles for each verb class. TF-IDF stands for term frequency and inverse document frequency. It is a statistical measurement that is usually used to measure how important a token or term is to a document in a collection of documents or corpus. In this case the terms we were measuring were the thematic role labels.

To get the term frequency for the thematic role labels, we went through each verb class and kept a count on the number of thematic roles seen. Additionally, we weighted the count of the thematic role by the verb class's average ranking correlation. The product of this was a map that mapped each thematic role label to a weighted count that represented term frequency. To get the document frequency we simply counted the number of verb classes that the thematic role appeared in. Using the term frequency and document frequency we calculated TF-IDF and sorted the thematic role labels from highest to lowest TF-IDF score. Having a higher score means that thematic role label is more important and is highly correlated to the verb classes that consistently have better embeddings. The results for both the Emory Word2Vec and DEP1 embeddings are in Table A.1 in the appendix.

In VerbNet the thematic role labels that are attributed to each verb class, are assigned a set of semantic restrictions. These restrictions constrain the types of thematic roles allowed by the arguments [11]. The same thematic role can have different sets of semantic restrictions for two different verb classes. For example both verb class wink-40.3.1 and the verb class consider-29.9 have the thematic role of Agent. However, the set of semantic restrictions on Agent for consider-29.9 is animate and organization. Whereas the set of semantic restrictions on Agent for wink-40.3.1 is recipient and concrete.

We did the same measure of TF-IDF on the semantic restrictions as we did for thematic role labels and the results are listed in Table A.2 in the appendix.

4.2.2 Result Analysis

It was very interesting to find that certain verb class consistently create better verb embeddings and that this does not change between different types of context windows. For this reason we wanted to look into the attributes of each verb classes such as their thematic roles and semantic restrictions to see if this gave any insight into why certain verb classes might do better. Using TF-IDF on both thematic roles and semantic restrictions allowed us to evaluate the importance of these attributes.

The results of TF-IDF for thematic roles in table A.1 show clearly that certain thematic roles are better than others and that this corresponds to the average verb ranking for a verb class. However, none of the thematic roles have an especially better or worse ranking. Plotting the values of this table, shown in Figure 4.2 in sorted order, shows a fairly linear trend with no thematic roles clustered around a particular value of TF-IDF.

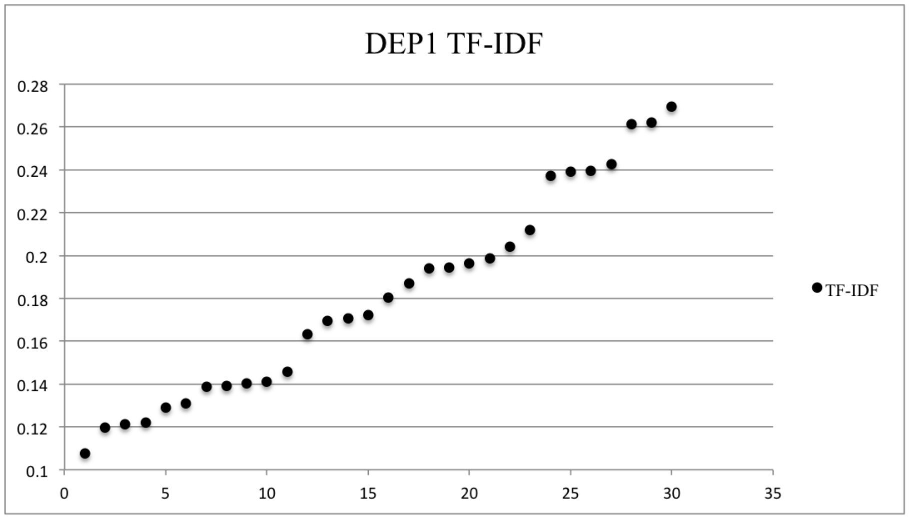


Figure 4.2: DEP1 Thematic Roles TF-IDF
X Axis: the thematic role id number
Y Axis: the TF-IDF score

The results of TF-IDF for semantic restrictions in table A.2 also show clearly that certain semantic restrictions are better than others. However unlike the thematic roles, certain semantic restriction are especially better than others. Plotting the values of the TF-IDF of the semantic restrictions, in sorted order, shows a fairly linear trend for the first 21 semantic restrictions. However, the top 8 semantic restrictions follow an exponential trend. This graph is shown in Figure 4.3. The plot for Emory W2V follows this same pattern and can be seen in Figure A.1 in the appendix. The top 6 restrictions are the same for both Emory W2V and DEP1, although they are in a different order. From this we can conclude that these verbs with these semantic restrictions are strongly correlated to have better embeddings.

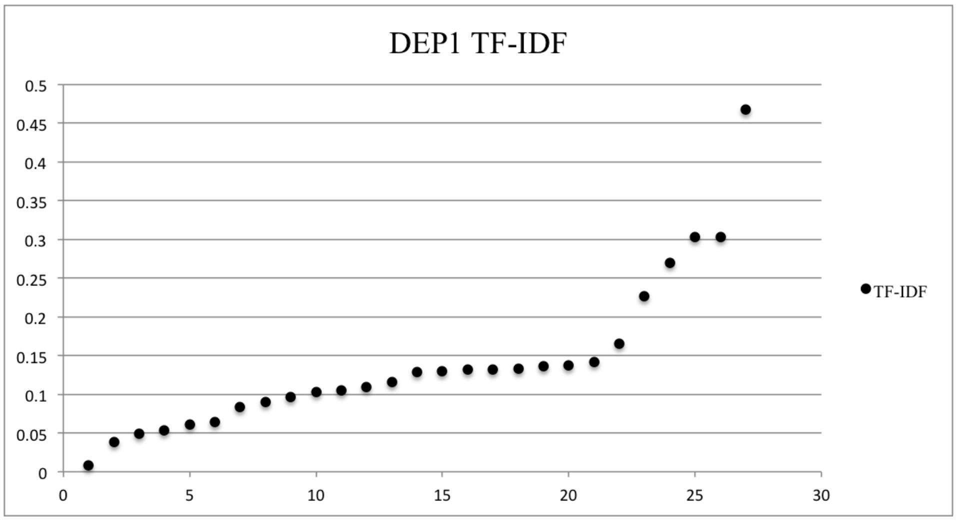


Figure 4.3: DEP1 Semantic Restrictions TF-IDF
X Axis: the semantic restriction id number
Y Axis: the TF-IDF score

4.3 Extrinsic Evaluations on Sentiment Analysis

Word embeddings are useful because they are used as features in NLP tasks. In order to highlight their usefulness and to show that our systems embeddings are useful and can outperform the previous Word2Vec embeddings, we used the embeddings in a sentiment analysis task. In this case we use the word vectors as direct input to a sentiment analysis system composed of a convolutional neural network. This system aims to create a representation of the sentiment of a sentence using the word embeddings. This takes our work on distributional semantics from the word level and raises it to the sentence level.

4.3.1 Background on Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a type of neural network that has one or more convolutional layers. Each of the convolutional layers of a CNN are usually followed by a subsampling layer and a CNN usually ends with one or more fully connected layers. CNNs have been traditionally used in the field of Computer Vision where the input to the network is an image. CNNs are recently gaining traction in the field of Natural Language Processing and have been used in semantic parsing and other tasks [13]. In this work, a CNN is applied to the word vectors contained in a sentence and produces a classification of sentiment for that sentence.

The weights between each convolutional layer of a CNN contains a set of filters, also known as kernels. Each kernel performs some convolution over a vector or matrix, in other words, each filter is convolved with the input. A convolution takes the set of values in the filter and computes the dot product between it and a window over the input. The filter is applied to all windows over the input. A basis term is usually added to the result and then that sum is usually multiplied by a nonlinear function resulting in feature Fi. A feature map F is the set of features Fi which are the results of the above operations over all windows of the input, where i denotes the window. If there are multiple filters then there will be multiple feature maps for the convolutional layer. Backpropagation is used to calculate the gradient on the filters of the convolutional layers and update the weights.

Subsampling layers always follow convolutional layers and are usually in the form of max, min or mean pooling. This type of layer does not do learning itself but is a way to reduce the feature space and to capture only the important features. Max pooling is when for each feature map F in the set of feature maps for a convolutional layer, only the maximum feature Fi is extracted and saved for further use. Mean pooling is similar but it takes the mean of all features instead of taking just the max. The smaller feature space creates the max pooling layer which is then passed to the fully connected layer.

Fully connected layers are composed of nodes that are connected to all the activation nodes in the previous layer. The CNN used in this work employs dropout regularization between the max pooling layer and the fully connected layer. For each training sample, dropout selects, by a defined probability p, which activation nodes to use and does not activate all the neurons [13]. The idea of dropout is that during a training sample it drops out some of the neurons, instead of using them all. Essentially dropout is used to stop the model from overfitting. During testing the network does not use dropout but will change the weights by a factor of the probability p [1]. Other types of regularization to reduce overfitting include L2 regularization which is usually the most common form of regularization.

4.3.2 System Overview

In this work a Convolutional Neural Network is used for sentiment analysis and classification. The input to the Convolutional Neural Network is the word embeddings for the words in the sentence that is being classified and the output is the sentiment class for the sentence. The network contains one convolutional layer with multiple filters, a max pooling layer following the convolutional layer, and one fully connected layer that is the last layer in the network. The size of the fully connected layer is based on the type of sentiment classification being done. Since the classification is binary into negative and positive sentiment, the layer has a size of two. The convolutional layer is fed the word vectors concatenated to form a two dimensional matrix. The output of the convolutional layer is multiple vectors, where each vector represents the feature map of a different kernel. The vectors are then combined into one vector by doing max pooling, which takes the max value from each feature map vector and puts that value in the new max pooling vector. The vector then is passed to a fully connected layer, during which, dropout regularization is used to reduce overfitting. An image of the architecture is shown in Figure 4.4.

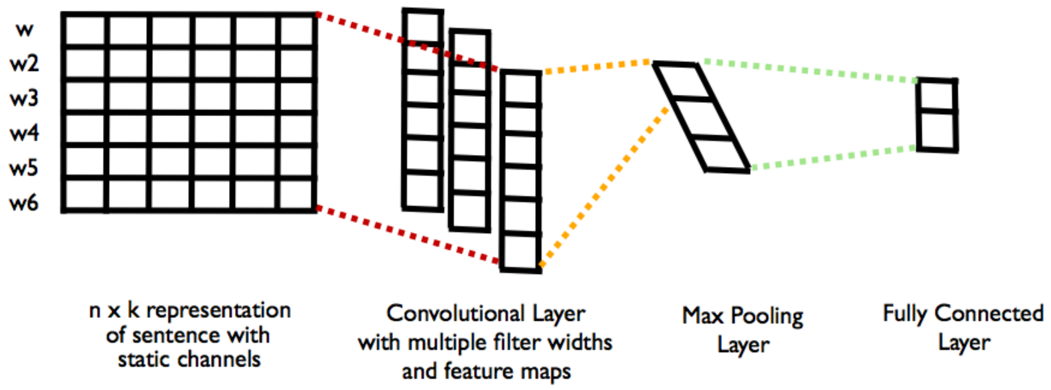

w
w2
w3
w4
w5
w6
n x k representation of sentence with static channels
Convolutional Layer with multiple filter widths and feature maps
Max Pooling Layer
Fully Connected Layer

Figure 4.4: Convolutional Neural Network Architecture
Shows the architecture of the CNN used in our system

We used the Theano framework [6] [8], which is a python based framework to create the neural network. We based our convolutional neural network system off the work from Yoon Kim [22]. For our sentiment analysis testing and training data we used Rotten Tomatoes movie reviews [15] [18] that we retrieved from the Kaggle Sentiment Analysis Challenge. The dataset originally had 5 categories, but for simplicity, we modified the dataset to only contain two categories of sentiment: positive and negative. There were 34345 negative sentences and 42133 positive sentences in the combined training and test data. We ran the both the Emory

W2V embeddings and all the structure based embeddings, that we created, through the CNN sentiment analysis system and found the results shown in Table 4.6 and a bar chart of the accuracy for the embeddings is shown in the appendix in Figure A.2.

	Accuracy
Emory W2V	0.8624373
DEP1	0.8814250
DEP1H	0.8867580
DEP12	0.8927635
DEP12H	0.8969352
DEP1SIB1	0.8910444
DEP1ALLSIB	0.8896382
DEP1SRLARG	0.8925598
DEP1SRLH	0.8997660

Table 4.6: Sentiment Analysis Accuracy

4.3.3 Result Analysis

From the accuracy scores of the sentiment analysis data, which is percentage of the number of correctly classified sentences, we can tell that the structure based embeddings clearly outperform the Emory W2V embeddings. This further establishes the point that structural based dependency embeddings are higher quality than the Emory W2V embeddings. In future work we would like to modify the system to run the 5 categories of sentiment that the dataset originally contained.

Chapter 5

Conclusions and Future Work

From the experiments done in this work we can see that there is clear differences across embeddings for different parts of speech. The distributional semantic theory that states that words are most similar to words that occur is similar contexts holds true. This work shows that the type of context chosen is extremely important when creating semantic representations of words. It also shows that the best context will actually vary depending on the part of speech of the word. In this process we created a sound method of evaluating and testing new word embeddings according to different parts of speech. These novel evaluation methods will be useful for future work on improving word embeddings. Additionally these evaluation methods allow us to examine the previously unexplored vector space of word embeddings.

The work also shows that choosing contexts based on deeper sentence structures, such as predicate structures and dependency structures, can yield better results then the original Word2Vec approach. In particular the evaluations of this work show that on average dependency structures are the best contextual map for semantics of words.

We also looked into subcategories within verbs. Within verbs we found that certain verb classes have a consistent trend of producing better embeddings. We were able to examine the attributes of all the verb classes and pick out that the attribute of semantic restrictions were a particularly good indicator of verb classes that tend to have better embeddings.

Moving forward this work can serve as a basis for examining different semantic contexts. Additionally this work can be useful for people who are working on different tasks that require certain parts of speech and they want to choose which embeddings to use. Future work would include creating an ensemble of the best vectors for each POS and running tests on that. Additionally it would be interesting to look at other subclasses within POS other than verbs.

Appendix A

Tables of Evaluation Results

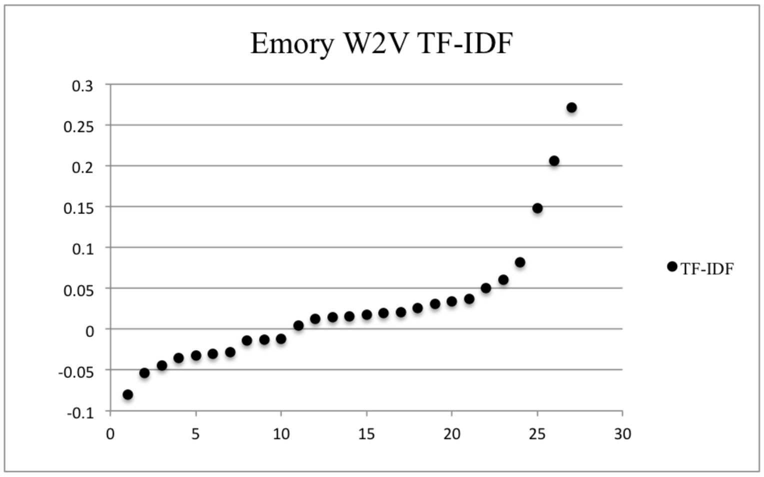


Figure A.1: W2V TF-IDF for Semantic Restrictions
X Axis: the semantic restriction id number
Y Axis: the TF-IDF score

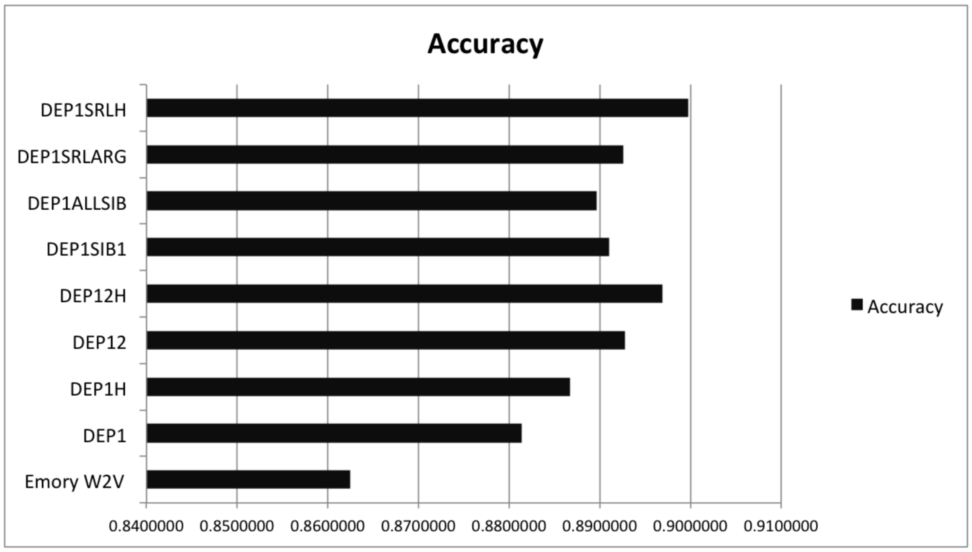


Figure A.2: Accuracy Bar Chart for Sentiment Analysis Task

DEP1		Emory W2V	
Thematic Role	TF-IDF	Thematic Role	TF-IDF
Location	0.107608684	Location	0.00876584
Destination	0.119610453	Destination	0.015258412
Patient	0.121426347	Patient	0.023213425
Result	0.122216405	Result	0.023778535
Instrument	0.129138923	Co-Patient	0.028328186
Value	0.131041313	Stimulus	0.03067366
Co-Patient	0.138650809	Initial_Location	0.032537165
Initial_Location	0.139321081	Instrument	0.032933738
Stimulus	0.140313702	Experiencer	0.035102587
Experiencer	0.141058461	Value	0.036818516
Trajectory	0.14588342	Extent	0.049864347
Agent	0.163313632	Pivot	0.050109701
Recipient	0.169545778	Agent	0.0580058
Product	0.170471465	Trajectory	0.059337197
Theme	0.172300911	Theme	0.063239456
Beneficiary	0.18038376	Recipient	0.068819099
Asset	0.187020566	Beneficiary	0.076501249
Pivot	0.194013264	Product	0.082942314
Time	0.194370835	Asset	0.088934328
Topic	0.196312019	Co-Agent	0.093085029
Material	0.198571887	Topic	0.096823725
Co-Agent	0.204286617	Material	0.098663215
Source	0.212100147	Time	0.099959924
Attribute	0.23705281	Source	0.10859664
Co-Theme	0.239254579	Attribute	0.116194637
Predicate	0.23961796	Co-Theme	0.128215774
Goal	0.242490007	Goal	0.130279908
Cause	0.261291256	Reflexive	0.138160557
Reflexive	0.262164058	Predicate	0.14018803
Extent	0.26936512	Cause	0.15957498

Table A.1: Thematic Roles TF-IDF

DEP1		Emory W2V	
Semantic Rest.	TF-IDF	Semantic Rest.	TF-IDF
animal	0.008069752	vehicle	-0.080535869
vehicle	0.038133761	animal	-0.05344451
force	0.049097646	substance	-0.044058901
refl	0.053354841	nonrigid	-0.035023411
body_part	0.061043103	solid	-0.032667689
nonrigid	0.064034098	force	-0.030145113
elongated	0.083371791	refl	-0.028468917
pointy	0.089672872	pointy	-0.013972301
substance	0.096195728	elongated	-0.012872145
comestible	0.103069985	body_part	-0.01196419
int_control	0.104793193	human	0.004031331
solid	0.109090853	plural	0.013075965
human	0.116119565	concrete	0.014488045
region	0.128640466	int_control	0.015938813
garment	0.130111673	region	0.017543195
machine	0.131590587	sound	0.019419068
communication	0.13172907	comestible	0.020619526
plural	0.132661374	location	0.025539132
location	0.13659233	machine	0.0311451
sound	0.137749398	communication	0.033841472
concrete	0.141796994	garment	0.037053907
currency	0.165268205	scalar	0.049864347
animate	0.226286459	currency	0.060949444
scalar	0.26936512	animate	0.082169937
time	0.302649297	organization	0.147867269
organization	0.303410729	time	0.20643542
abstract	0.467810365	abstract	0.271332069

Table A.2: Semantic Restrictions TF-IDF

Appendix B

Semantic Roles and Their Functions

- Actor: used for some communication classes when both arguments can be considered symmetrical (pseudo-agents).
- Agent: generally a human or an animate subject. Used mostly as a volitional agent, but also used in VerbNet for internally controlled subjects such as forces and machines.
- Asset: used for the Sum of Money Alternation
- Attribute: attribute of Patient/Theme refers to a quality of something that is being changed
- Beneficiary: the entity that benefits from some action Generally introduced by the preposition 'for', or double object variant in the benefactive alternation.
- Cause: used mostly by classes involving Psychological Verbs and Verbs Involving the Body.
- Location, Destination, Source: used for spatial locations.
- Destination: end point of the motion, or direction towards which the motion is directed.
- Source: start point of the motion. Usually introduced by a source prepositional phrase (mostly headed by 'from' or 'out of')

- Location: underspecified destination, source, or place, in general introduced by a locative or path prepositional phrase.
- Experiencer: used for a participant that is aware or experiencing something.
- Extent: to specify the range or degree of change
- Instrument: used for objects (or forces) that come in contact with an object and cause some change in them. Generally introduced by a 'with' prepositional phrase.
- Material and Product: used by classes from Verbs of Creation and Transformation that allow for the Material/Product Alternation.
- Material: start point of transformation.
- Product: end result of transformation.
- Patient: used for participants that are undergoing a process or that have been affected in some way.
- Predicate: used for classes with a predicative complement.
- Recipient: target of the transfer
- Stimulus: used by Verbs of Perception for events or objects that elicit some response from an experiencer. This role usually imposes no restrictions.
- Theme: used for participants in a location or undergoing a change of location
- Time: class to express time.

- Topic: topic of communication verbs to handle theme/topic of the conversation or transfer of message.

Bibliography

[1] A. Gibiansky, *Convolutional Neural Networks*, http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/, 2014.

[2] A. Ng, J. Ngiam, C. Foo, Y. Mai, C. Suen, A. Coates, A. Maas, A. Hannun, B. Huval, T. Wang, S. Tandon *Convolutional Neural Network* UFLDL Tutorial, Stanford University, http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/,2013.

[3] C. Leacock and M. Chodorow, *Combining local context and WordNet similarity for word sense identification*, WordNet: An electronic lexical database, Vol. 49, No. 2, pp. 265-283, 1998.

[4] D. Lin, *An information-theoretic definition of similarity.*, ICML, Vol. 98, pp. 296-304, 1998.

[5] Daniel Jurafsky & James H. Martin, *Speech and Language Processing: Semantic Role Labeling*, Ch. 22, pp. 1-23, 2015.

[6] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley and Y. Bengio, *Theano: new features and speed improvements*, NIPS deep learning workshop, 2012.

[7] Hideki Shima, *Word Similarity For Java* Carnegie Mellon University, https://code.google.com/archive/p/ws4j/, 2015.

[8] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley and Y. Bengio, *Theano: A CPU and GPU Math Expression Compiler*, Proceedings of the Python for Scientific Computing Conference (SciPy), 2010.

[9] J. Choi, *EmoryNLP NLP4J*, Emory University, Version 1.0.0, 2016.

[10] J. Nivre and S. Kubler *Dependeny Parsing Tutorial* COLING-ACL, Sydeny 2006.

[11] K. Kipper, A. Korhonen, N. Ryant, and M. Palmer, *A large-scale classification of English verbs*, Language Resources and Evaluation, Springer, Vol. 42, No. 1, pp. 21-40, 2008.

[12] Mark Finlayson, *JVerbnet.* Massachusetts Institute of Technology, https://code.google.com/archive/p/ws4j/, 2012.

[13] N. Srivastava, G. Hinton, Geoffrey, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, The Journal of Machine Learning Research, JMLR.org, Vol. 15, No. 1, pp. 1929-1958, 2014.

[14] O. Levy and Y. Goldberg, *Dependency Based Word Embeddings* Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Vol. 2, pp. 302-308, 2014.

[15] Pang and L. Lee, *Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales* ACL, pp. 115124, 2005.

[16] Princeton University, *About WordNet.* WordNet, Princeton University, http://wordnet.princeton.edu, 2010.

[17] Q. Le and T. Mikolov, *Distributed representations of sentences and documents*, arXiv preprint arXiv:1405.4053, 2014.

[18] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, and C. Potts, *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank*, Conference on Empirical Methods in Natural Language Processing (EMNLP), 2013.

[19] Statistics Solutions, *Correlation (Pearson, Kendall, Spearman)* Statistics Solutions, http://www.statisticssolutions.com/correlation-pearson-kendall-spearman/.

[20] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, *Distributed representations of words and phrases and their compositionality*, In Advances in Neural Information Processing Systems, NIPS, 2013.

[21] T. Mikolov, K. Chen, G. Corrado and J. Dean. *Efficient estimation of word representations in vector space*, ICLR, 2013.

[22] Y. Kim *Convolutional Neural Networks for Sentence Classification*, arXiv preprint arXiv:1408.5882, 2014.

[23] Z. Harris, *Distributional structure*, Word, Taylor & Francis, Vol. 10, No. 2-3, pp. 146-162, 1954.

[24] Z. Wup and M. Palmer, *Verbs semantics and lexical selection*, Association for Com-

putational Linguistics, Proceedings of the 32nd annual meeting on Association for Computational Linguistics, pp. 133-138, 1994.