Categorical Evaluation for Advanced Distributional Semantic Models

by

Andrew Reid Kilgore

Jinho D. Choi, Ph.D.
Adviser

Department of Mathematics And Computer Science

Jinho D. Choi, Ph.D.
Adviser

Phillip Wolff, Ph.D.
Committee Member

Valerie Summet, Ph.D.
Committee Member

2016

Categorical Evaluation for
Advanced Distributional Semantic Models

by

Andrew Reid Kilgore

Jinho D. Choi, Ph.D.

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 Business Administration with Honors

Department of Mathematics and Computer Science

2016

Abstract

Distributional Semantic word representation allows Natural Language Processing systems to extract and model an immense amount of information about a language. This technique maps words into a high dimensional continuous space through the use of a single-layer neural network. This process has allowed for advances in many Natural Language Processing research areas and tasks. These representation models are evaluated with the use of analogy tests, questions of the form "*If a is to a' then b is to what?*" are answered by composing multiple word vectors and searching the vector space.

During the neural network training process, each word is examined as a member of its context. Generally, a word's context is considered to be the elements adjacent to it within a sentence. While some work has been conducted examining the effect of expanding this definition, very little exploration has been done in this area. Further, no inquiry has been conducted as to the specific linguistic competencies of these models or whether modifying their contexts impacts the information they extract.

In this paper we propose a thorough analysis of the various lexical and grammatical competencies of distributional semantic models. We aim to leverage analogy tests to evaluate the most advanced distributional model across 14 different types of linguistic relationships. With this information we will then be able to investigate as to whether modifying the training context renders any differences in quality across any of these categories. Ideally we will be able to identify approaches to training that increase precision in some specific linguistic categories, which will allow us to investigate whether these improvements can be combined by joining the information used in different training approaches to build a single, improved, model.

Categorical Evaluation for
Advanced Distributional Semantic Models

by

Andrew Reid Kilgore

Jinho D. Choi, Ph.D.
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 Business Administration with Honors

Department of Mathematics and Computer Science

2016

Acknowledgements

First, I would like to thank my family for their ceaseless love and support.

Thank you to my adviser, Dr. Jinho Choi for providing guidance, encouragement and resources throughout this entire project. This work would not have been possible without his direction.

Thank you to Dr. Valerie Summet and Dr. Phillip Wolff for their time, expertise and advice both throughout my career at Emory and during the process of writing this thesis.

Lastly, thank you to the entire Department of Mathematics and Computer Science of Emory University.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The field of Natural Language Processing is generally concerned with the computational extraction and manipulation of language information. In order to extract as much information as possible, NLP systems find it useful to represent words in different ways. The obvious, and most common, method is to represent each word as an index into a vocabulary. While this approach is useful for many basic tasks it imposes a limit on the complexity of information that can be extracted from raw text. Distributional semantics gives us a new approach, statistically analyzing word occurrence to develop a better understanding of the entire language. Distributional semantics recently have introduced methods to map words to high-dimensional vectors which can capture an immense amount of syntactic information. These vectors are also referred to as word embeddings. The impact of advances in distributional semantics, particularly that of word embeddings generated with neural network models, can not be understated.

The use of high-quality continuous word embeddings has been one of the most useful recent advances in Natural Language Processing. Models can be built to contain an incredible amount of information about each individual word, particularly in relation to other words. One of the most interesting benefits of this approach is the ability of these models to perform logical operations on words. Notably, this means that many computational linguistic models can now represent relationships between words and solve basic analogy problems of the form *a is to a' as b is to blank*. This capability is a direct result of the continuous nature of word embeddings. The offset between two words a and a' can be interpreted as their syntactic difference, it therefore follows that the relationship between any two words can be extracted in this way. This offset can then be applied to some other word b to find the appropriate b', if it exists. Analogy tests of this form have been found to be very effective measures of overall word embedding quality[12].

In this paper we analyze the quality of word embedding models across different syntactic categories. We accomplish this by utilizing a relationship extraction technique to perform analogy tests across a diverse test set composed of questions in 14 different linguistic categories. Additionally, we introduce methods to create a variety of alternative word embeddings utilizing different forms of syntactic and semantic information. These new embeddings are used in conjunction with our test sets to determine a link between specific linguistic information leveraged in training and particular competencies of word embedding models.

In order to achieve these results we introduce an expanded implementation of Mikolov's Word2Vec[16] that creates word embeddings based on arbitrarily defined linguistic information, a multithreaded analogy testing framework to massively decrease the time taken by this task and demonstrate an analysis methodology to help determine the optimal embedding training method for particular tasks. Additionally we provide novel set of word embedding models trained based on a variety of different linguistic information.

1.1 Thesis Statement

By analyzing lexical and grammatical competencies across a set of word embedding models, including a set created by using syntactic information, we intend to discover whether current techniques insufficiently model particular aspects of language. Further, we intend to explore whether linguistic information can be used to guide model training in a way that combines and enhances model competencies across syntactic categories. We finally introduce a system capable of selecting the best possible model to use at run-time, thereby utilizing unique or difficult-to-replicate model competencies.

Chapter 2

Background

The field of Natural Language Processing (NLP) is largely concerned with the development of algorithms that can extract information from human language. Problems in this field have focuses ranging from semantic to purely linguistic tasks, including examples such as sentiment analysis and dependency parsing respectively. NLP systems generally utilize at least one word representation method in their approaches to these problems. Word representation can be achieved in many different ways, from simply using text representation to leveraging high dimensional vectors.

The most influential recent advance in word representation is the success of distributional semantics, which represents words using high-dimensional vectors. Distributional representation entails mathematically mapping words into a high dimensional space, generally accomplished using a neural network. The end result of this process is usually referred to either as a word embedding model or distributional semantic model. This representation of words in a continuous space has proven immensely useful for a large variety of NLP tasks.

Building word embedding models has been most effectively achieved as the result of a neural network training task. Specifically, Mikolov's neural network training method is considered to be the most advanced way to generate word embeddings[16][7]. They present a neural network which trains using a large raw text corpus to perform a slot-filling task.

Given a series of words $x_{i-n}, x_{i-n-1}, ..., x_{i-1}, x_{i+1}, \ldots x_{i+n}$ called C, where i is the position of the training word in a sentence and n is the context window size, the task is to select from the vocabulary some word y to fill position i such that:

$$\arg\max_y \; p(y|C)$$

which is equilvalent to:

$$\arg\max_y \; p(y|x_{i-n}, x_{i-n-1}, ..., x_{i-1}, x_{i+1}, ..., x_{i+n})$$

This approach extracts an incredible amount of information from language, particularly in terms of relationships between different words. The success of this approach is often explained with the Distributional Hypothesis. This hypothesis states that the frequency with which words co-occur, or do not co-occur, with each other contains all of the information about a language. The basis of this argument is that words that frequently appear together are similar, therefore knowing the total frequencies with which a given word x appears with every other word should render a complete definition the meaning of x[8].

One of the problems of distributional semantics is model evaluation. There is no easily determined ground truth for a vectorized representations of words. Mikolov presented a framework to solve this challenge leveraging the unique capability of word embeddings to capture the relationship between two words. A set of questions of the form "*a is to a′ as b is to blank*" are posed to the word embedding model, which is tasked to find the best possible word to fill in the blank.[17] Analogy tests are one of the most frequently used methods to evaluate word embedding quality.

2.1 Word Representation

One of the most useful basic tools of NLP is word representation. In human language, words are generally either represented as audio signals or printed text. Humans are trained to understand and process language represented in these and other forms, but machines have no innate ability to achieve this same result. Methods to computationally represent words include tokenization (indexing into a text vocabulary), Brown Clustering[2] (associating words by the amount of information they share under an N-gram model), Latent Semantic Analysis[5] (term frequency and document similarity) and Distributional Word Embeddings (mapping words onto a high-dimensional continuous space). Distributional semantics methods are the most advanced techniques available for word representation. For this reason we focus our discussion of word representation on distributional semantic models.

2.1.1 Word Embeddings

High-dimensional word embeddings have recently been widely utilized to better represent semantic meaning in language. Word embeddings, which map words into high-dimensional vector spaces, allow for many novel approaches to language modeling.

The most influential neural network architecture for distributional word representation is Mikolov's SkipGram with negative sampling[16]. This approach utilizes word tokenization to train a single-layer neural network to efficiently generate high-quality word embeddings. The original C implementation of this approach can be found in the Word2Vec package[16], but many other implementations exist in various languages such as the Python Gensim package[14] and the Java EmoryNLP package[3].

Distributional Semantic Hypotheses

Distributional semantics is concerned with the representation of words and meaning in a high dimensional vector space. The work in this field generally has a theoretical basis in the following hypotheses which posit the possibility of learning language meaning through purely statistical measures. These hypotheses provide the underpinnings of the findings of this paper and an explanation for the effectiveness of high dimensional continuous word embeddings in a variety of tasks.

Statistical Semantics Hypothesis The statistical semantics hypothesis states that the meaning behind a sentence can be determined by observing statistical patterns of natural language usage. Practically, it implies that if vectorized representations of two words are similar then those words will likely have similar meanings. This hypothesis is a more formal and broader statement of the following theories.[18]

Bag of Words Hypothesis Given a query Q, a document D is likely to be found to be relevant to Q if their word frequencies are similar. The more similar the word frequencies are between Q and D the more likely it is for D to be a relevant answer to Q. More specifically, the two items are thought to be similar in meaning if their word frequencies are similar.[15][18]

Distributional Hypothesis The distributional hypothesis posits that all the information about a word in a language can be determined by its statistical relationship with other words. Less formally, Firth put forward the idea that *"a word is characterized by the company it keeps"*[6]. The implication here is that words are similar to other words they cooccur with and dissimilar from words that they do not cooccur with. This means that, for a given word, the exact meaning of that word can be captured by the the frequency with which it word appears, or does not appear, with every other word in the language. Aggregated over a sufficiently large corpus, it follows that such a method could capture all semantic meaning from an entire language.[8]

Applications

Possibly the most important aspect of modern distributional semantics is its incredible effectiveness in a wide range of NLP tasks. Many different tasks, including Named Entity Recognition, Question Answering and Dependency parsing, have been dramatically improved by their use of distributional semantic models.

2.2 Analogy Tests

Analogy testing is one of the most well-regarded word embedding quality assessment frameworks currently used. This method is generally only used to show a single score as the total competency of the model, without much linguistic interpretation or insight. We believe analogy testing has an unexplored potential to provide analysis of very specific competencies and qualities of language models.

2.2.1 Vector Offset

Possibly the most useful aspect of distributional semantic models is their ability to capture the relationship between two words. This is achieved by taking the offset between two vectors. For instance, extracting the offset between *vector(Man)* and *vector(Woman)* gives the relationship of gender. This is accomplished by the following formula:

$$relationship(a : a') = \vec{a'} - \vec{a}$$

Where a and a' are a pair of words and $\vec{a'}$ and $\vec{a}$ are their vector representations. This offset can be taken and applied to other words, say *vector(Boy)* to find the word embedding representation of the word *Girl*. More generally, given a pair a and a' and some other word b we can apply $relationship(a : a')$ to b with:

$$\vec{b'} = relationship(a : a') + \vec{b}$$

Therefore:

$$\vec{b'} = \vec{a'} - \vec{a} + \vec{b}$$

This implies that the relationship between any pair of words should be found as a constant offset between those words, and that this offset will be the same for all other similar pairwise relationships.

2.2.2 Analogies

Analogies themselves obey the following pattern: given two similar pairs of words *a,a'* and *b,b'* the relationship between *a* and *a'* is the same as that between *b* and *b'*. This relationship is used to create a slot-filling task: given *a*, *a'* and *b* the goal is to pick the best possible value for *b'*.

This task can be translated for distributional semantic models easily. We simply take the vector representations of *a,a'* and *b*, use the following formula[12]:

$$\arg\max_{\vec{x}} \ \textsc{CosineSimilarity}(\vec{x}, \vec{a'} - \vec{a} + \vec{b})$$

where $\vec{x}$ should be the vector representation of b'.

The results from this task are often surprising in how well they expose semantic meaning. One common example is that, on sufficiently high quality embeddings, it is possible to take x *= vector(King) - vector(man) + vector(woman)* such that searching the vector space for the most similar word to x gives *vector(Queen).* This process uses the continuous nature of word embedding models to extract a relationship, in this case gender, from two words and then apply it to a third word to complete the analogy.

2.2.3 Syntactic Test Set

We base our evaluation on a collection of tests from Mikolov, with a total of 19544 individual tests[17]. We broke the tests down into fine-grained categories relating to different parts of speech and semantic meanings. Each test is of the form *"a is to b as b is to x"* where the task is to find the best possible value for x.

The test set includes both lexical and grammatical tests. The lexical categories that it tests are common world capitals (Madrid is to Spain as Beijing is to China), all world capitals (a superset of the common world capitals), city in state (Atlanta is to Georgia as Chicago is to Illinois), currency (Mexico is to Peso as England is to Pound) and family (Mother is to Daughter as Father is to Son). The grammatically focused categories are adjective-to-adverb, opposites, comparatives, superlatives, present-participles, nationality-adjectives, past tenses, plurals and plural verbs.

2.2.4 Test Set Customization

To more properly analyze results from these test sets we reorganized them for our uses. The grammatical category of opposites was moved to the lexical category, while the remaining grammar-focused test sets were merged to form a grammatical test set. This left us with a final lexical test set and grammatical test set, each split into seven subcategories. Additionally, upon examining the family test cases it turned out that every single test only actually required a model to determine difference in gender, not actual family relations. For example, there were tests along the lines of "brother is to sister as son is to daughter", but none along the lines of "son is to father as father is to grandfather". To enhance the clarity of our findings we refer to the family test set as the gender test set from this point on. Examples and categorizations of these test sets can be found in Table 2.1.

Table 2.1: Analogy Examples

Category	**Type**	**Example**
capital-common-countries	Lexical	England is to London
capital-world	Lexical	Nigeria is to Abuja
city-in-state	Lexical	Los Angeles is to California
currency	Lexical	England is to Pound
gender	Lexical	King is to Queen
adjective-to-adverb	Grammatical	Amazing is to Amazingly
opposite	Lexical	Hot is to Cold
comparative	Grammatical	Warm is to Warmer
superlative	Grammatical	Loud is to Loudest
present-participle	Grammatical	Coding is to Code
nationality-adjective	Lexical	American is to America
past-tense	Grammatical	Dancing is to Danced
plural	Grammatical	City is to Cities
plural-verbs	Grammatical	Describe is to Describes

2.2.5 Scoring - Precision and Recall

Precision and recall are measures generally used for information retrieval with binary classification and pattern recognition. The premise is that a model may retrieve information from four different general categories: correct and relevant, incorrect and relevant, correct and irrelevant and correct and irrelevant. Precision is a fraction of how much retrieved information is relevant. Recall is a fraction of how much of the total relevant information was returned. This difference becomes very important in understanding the quality of a given model or approach. Analogy Test results are given as the precision of the model, essentially the number of correct answers returned as a percentage of all of the answers returned as seen here:

$$\textsc{Precision} = \frac{\textsc{Correct Analogies}}{\textsc{Answerable Analogies}}$$

We use this measure because recall in this case is linked strongly to the vocabulary of the model, which is predominantly a function of the corpus used. Additionally, there is no current method to register that a word embedding model is attempting to return an out-of-vocabulary word. In this work "score" and "accuracy" may be used interchangeably in place of "precision".

2.3 Linguistic Structure

2.3.1 Dependency Structure

The syntactic theory of dependency is the idea that a sentence can be modeled using directed binary relations between words. Using dependency, words can be organized into a graph by treating each word as a node and their relations as arcs. Each word in a dependency graph may have multiple outgoing arcs but only one incoming arc. This means that dependency graphs naturally organize themselves into trees, where a word has a single head and may have any number of children. Dependency is a one-to-one relation, any given word in a sentence maps to exactly one member of the resulting graph. To keep these rules consistent, a *false head* node is usually introduced, whose only rule is to provide a head to the root of the dependency tree. For example, in Figure 2.1, *"bought"* is the head of *"He"*, *"car"*, and *"yesterday"*. The head of *"bought"* is the *false head*. The siblings of *"car"* are *"He"* and *"yesterday"*.

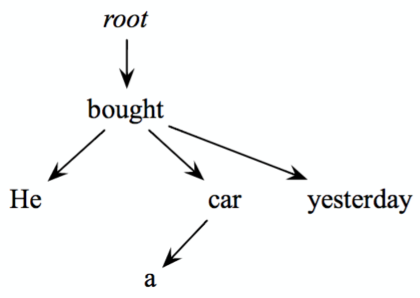


Figure 2.1: Dependency Tree[4]

2.3.2 Predicate Argument Structure

Predicate argument structure is largely concerned with the role of semantic arguments in language. Semantic arguments are words in a sentence that are associated with the predicate or a verb. Each argument is classified by its role and the meaning that it adds to the sentence. The semantic role label is the label attributed to a specific relation between a word and its semantic argument. For example, in the sentence "*I saw a car*", the verb *"saw"* takes arguments *"I"* and *"car"*. This kind of semantic abstraction provides very specific information about the sentence as the role of semantic arguments are very strictly defined.

As illustrated in Figure 2.2, each verb only accepts very specific arguments. For instance the verb *"open"* will only accept arguments that provide either an *opener*, *thing opened*, *instrument* or a *benefactor*. Given a sentence *"Mary opened the bottle with a corkscrew."* the words *"Mary"*, *"bottle"* and *"corkscrew"* would serve as ARG0, ARG1 and ARG2 respectively. Not all arguments must be present every time a verb or predicate is used.

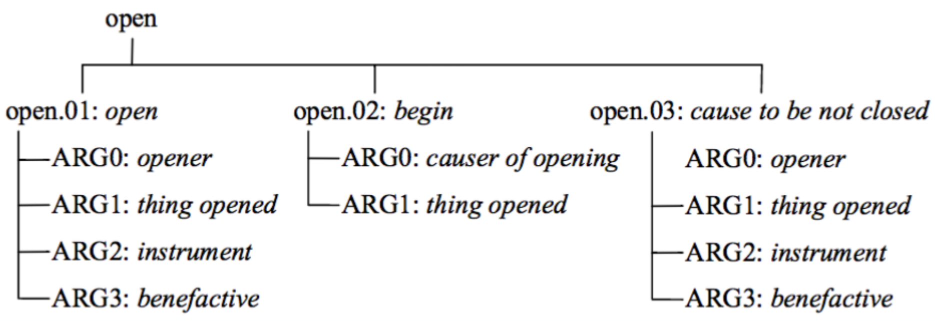


Figure 2.2: Possible semantic role arguments for "open" words.[4]

2.3.3 Morphemes

A morpheme is the linguistic atom, it is a language's smallest possible grammatical unit that still contains meaning while possibly standing alone (but not necessarily). Broadly speaking, a single morpheme can be classified as a member of one of two categories: Inflectional or Derivational. Membership in either group is based on the effect a morpheme has on the word that it is paired with and in what manner it changes that word's original meaning.

Inflectional Morphemes

Inflectional morphemes do not alter the part of speech of any word they are paired with. For nouns, pronouns and adjectives an inflectional morpheme can change a word's number, gender or case. In the case of verbs, this kind of morpheme might change tense, mood, number, person or aspect. For instance, adding -*ed* to the word *play* to form *played* and adding -*s* to *run* to form the word *runs* are both examples of inflectional morphemes.

Derivational Morphemes

Derivational morphemes, on the other hand, affect the actual meaning of a word in a much more pronounced way. Generally speaking, a derivational morpheme will either change the part of speech or semantic meaning. Adding -*en* to *dark* to form the word *darken* changes an adjective to a verb. Alternatively, adding -*un* to the word *helpful* renders the word *unhelpful*, thereby reversing the semantic meaning of the original word.

2.4 Neural Network Models

The Feedforward Neural Network Language Model and the Recurrent Neural Network Language Model were two of the earlier proposed architectures to create continuous word embeddings leveraging machine learning techniques. They both share a significant training complexity, inspiring advances such as utilizing hierarchical versions of softmax as well as the Bag of Words and SkipGram architectures.

2.4.1 NNLM

The representation of words in a continuous vector space has been achieved in various forms for several years. One of the earliest successful proposals that incorporated a neural network was the Feedforward Neural Net Language Model (NNLM). This network is comprised of input, projection, hidden and output layers. The computational complexity of this model can quickly become massive, inspiring various improvements[1].

2.4.2 RNNLM

The feedforward NNLM is limited in the need to specify, and therefore optimize for, context training size as well as the theoretical limits on its capability to learn complex patterns. This problem is rooted in the seeming lack of memory that a model has, the output from any previous iteration has no real effect on future outputs. In order to compensate for these problems, the Recurrent Neural Network Language Model (RNNLM) was introduced[11][10]. These networks are also composed of an input, hidden and output layer, but with a key difference. Recurrent Neural Networks pass their output back to the hidden layer while accepting new input. Each iteration is now a step forward in time, where the Network receives information from previous epochs at each time $t > 0$. This creates a weak form of memory inside of the neural network, later improved upon by the Long Short Term Memory neural network[9]. This RNN forgoes the projection layer of the NNLM, only using an input, hidden and output layer.

2.4.3 Word2Vec

While Mikolov et al. draw inspiration from NNLM and RNNLM, their Word2Vec model is built upon a more efficient architecture, of which there exist two variants. The effectiveness, and subsequent popularity, of these architectures is due in part to its use of negative sampling and very efficient training[16]. These architectures are visualized in Figure 2.3.

SkipGram The SkipGram architecture is a neural network model with a single hidden layer. It trains by taking each word x in each sentence and predicting what other words are adjacent to x. For our research we also leverage the SkipGram architecture in our generation

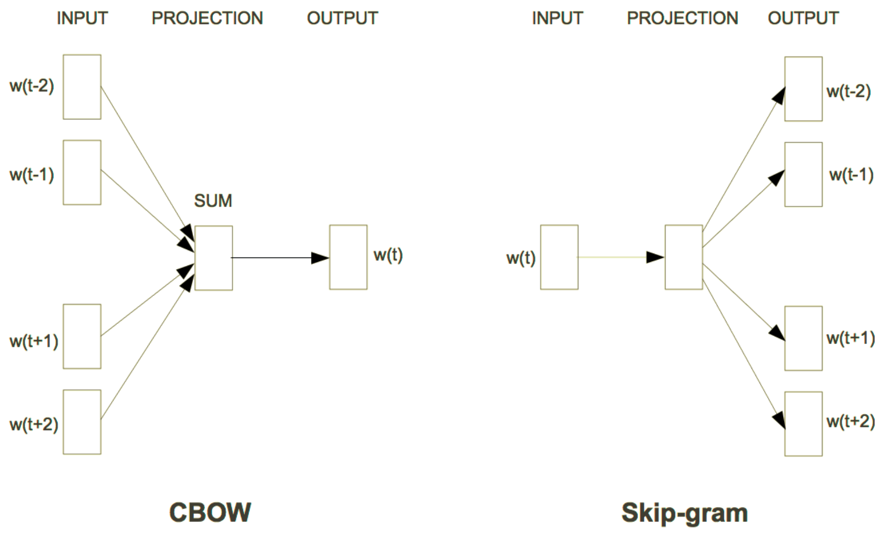


Figure 2.3: SkipGram and Continuous Bag-of-Words Architectures[17].

of word embedding models. Given some text corpus we take each word w and its context in order to examine the conditional probability of each word appearing with its context. Given training words $w_1, w_2, ..., w_T$ and a context size c SkipGram maximizes the following:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j<c, j\neq 0}\log p(w_{t+j}|w_t)$$

Continuous Bag-of-Words The reverse training task, predicting a single word from its context, is the continuous Bag-of-Words model. The goal of continuous Bag-of-Words is to take any given context and predict the missing word.[16].

Complexity The training complexity of both the SkipGram and Continuous Bag-of-Words architectures is

$$O = E \times T \times Q$$

where T is the number of words in the corpus, E is the number of training iterations and Q is the compexity of a single training iteration. Q is defined differently for each architecture.

For SkipGram, Q is:

$$Q = C \times (D + (D \times \log_2(V))$$

where C is the context size, D is the size of the hidden layer and V is the number of words in the vocabulary. The complexity of the models we use in this work is largely the same, where we vary C based on the current experiment.

2.4.4 Contexts

Word2Vec trains each word in a corpus differently based on its context. In this case, the context of a word is defined as whichever words are closest to it within the sentence. The SkipGram model is trained by maximizing the likelihood that, given a word, its context can be found. One problem with this is that important semantic meaning is often spread out distantly through a sentence, meaning that it is possible that related word pairs may not be trained together.

Arbitrary and Dependency Contexts

Arbitrary contexts have been used in order to explore different possible information extraction strategies for word embedding generation. The abstraction of this training method allows us to redefine what the context of a word consists of. For instance, the dependency structure of a sentence contains a huge amount of information regarding both the meaning of the sentence as a whole and of each individual word. Levy and Goldberg introduce this approach by training each word against its dependency head and the syntactic dependency label connecting the two tokens.[13] The major advantage of this approach is that it intelligently selects the context based on words that interact within the sentence instead of having to guess a window size. This increases the likelihood that words selected as context will provide valuable information, especially for tokens that would have been too distant from each other to be included in their respective SkipGram contexts.

Levy and Goldberg only attempt using the dependency head of each word as context. There is, however, a rich amount of syntactic information readily available to improve the training contexts. Mikolov recognizes that the SkipGram and Bag-of-Words architectures are limited by how quickly complexity increases as their context window increases. It is put forward that this potential loss of information is somewhat inconsequential as “distance words are usually less related to the current word than those close to it”[17]. This may or may not be true, but using expanded contexts that leverage syntactic information could greatly improve training by creating even more relevant context windows. In turn this should improve the quality of word embedding models while also decreasing the computational complexity from using large contexts.

Chapter 3

Approach

3.1 Corpus

We created our word embedding models using the 2015 dump of Wikipedia, a freely available dataset, as well as the New York Times news corpus. We preprocessed both of these corpuses using the EmoryNLP package[3] in order to obtain specific linguistic information to use in our training models including dependency and semantic role relations. The combination of these sources gives a total of 6,406,418,032 words, including 1,531,163 unique words.

3.2 Syntactic Contexts

In order to investigate the linguistic competencies gained by a word embedding model trained on different sets of information, we first extended the Python Gensim package[14] to train each word on an arbitrarily defined context. Eventually we also implemented arbitrary contexts in the Java EmoryNLP package in order to take advantage of its significantly faster training speed as well as its advanced dependency parsing library. All results presented are based off of our results from the modified EmoryNLP package.

Our corpus is first processed for dependency and semantic role labels. This processed corpus can later be used to train additional models. Each word is treated as a node with information including word form, dependents, dependency head as well as semantic role arcs and labels. Our training method makes heavy use of this dependency graph, traversing it locally to build most word embedding contexts. Crucially, we replace the windowed context training phase of Mikolov's Word2Vec[16] with a new context generating phase. The context of a word is generated based on linguistic information passed through to the main neural network training algorithm, thereby allowing new models to be dynamically and easily created.

The linguistic relationships we chose to explore at first were that of each given word's siblings, dependents and semantic head.

Refer to Appendix B for visualizations of each approach.

3.2.1 First Order Dependency (dep1)

The First Order Dependency model (dep1) made use of dependency structure, specifically the children of a particular node. For a word x in a sentence, any words with incoming links that originated at x were used as the context to train x.

3.2.2 Semantic Role Label Head (srl1)

Under the semantic role label head training method, each word is trained against any other word in the sentence that uses it as an argument. Therefore, in the sentence *"I bought a car."*, the word *I* would be trained against the word *bought*, as would the word *car*. The major drawback of this is data scarcity, very few words are semantic heads.

3.2.3 Closest Dependency Siblings (sib1)

Closet dependency siblings also takes advantage of dependency structure. For a given word x in a sentence, the context of that word is constructed by taking the dependency children of x's head which are adjacent to x. The neural network then attempts to determine the best possible missing sibling in this context set.

3.2.4 First and Second Closest Siblings (sib2)

This model is built in the same way as $sib1$, but the second closest siblings of the given word are also used to build its context.

3.3 Composite Models

After building this set of models leveraging basic syntactic information we decided to explore whether a more complex and linguistically-rich approach to training would improve the quality of our distributional representations. We combined and extended the training contexts of several of our basic syntactic models in order to determine whether we could effectively combine the information learned by different models.

3.3.1 All Siblings (allsib)

This model further explores the idea that valuable information is contained in knowing which words share a common head, while also trying to mitigate the problem of having too small of a context as much as possible.

3.3.2 Second Order Dependency (dep2)

This training method is most similar to first order dependency, but it also includes the dependents of all dependents of the given word. This provides the model with a greater understanding of the dependency structure of each sentence that it trains on.

3.3.3 Second Order Dependency with Head (dep2h)

This method is identical to second order dependency, but with the head of the given word added to the context.

3.3.4 Siblings with Dependents

In order to examine the effect of building contexts from those of smaller syntactic models we tried three different approaches that took disjoint contexts from previous models and combined them.

First Order Dependency with Closest Siblings (sib1dep1)

The context for this model is built by combing first order dependency with closest siblings.

First Siblings and First Order Dependency (sib2dep1)

This model is similar to $sib1dep1$ but with the addition of the second closest siblings of each word to its context.

First Siblings and Second Order Dependency (sib2dep1) This model is an attempt to combine the contexts of $sib1$ and $dep2$.

3.4 Ensemble Models

In order to maximize the information available in any given Natural Language processing task, we devised a method to create ensemble models from multiple different word embeddings with differing linguistic competencies. Our goal was to be able to pair distributional representations that performed well in different categories and then dynamically switch between them as the current task demanded.

In order to build an ensemble model we first established techniques to select which word embeddings would be included in the model. Next we created a new analogy testing framework to support evaluating a model that dynamically selects which word embeddings to use for each individual task. While our implementation is specific to analogy tests, the principle pre-processing technique can easily be extended to any other natural language processing task.

3.4.1 Model Inclusion

We selected models to use in our ensemble model in three different ways, by sum of ranking score, sum of categorical accuracy score, and our maximum information selection process which selects models by the amount of new and unique information they contribute. The goal was to maximize the competency with which the ensemble model could perform on any given natural language processing task where the linguistic category of the task was known or detectable given an arbitrary set of word embedding models. In the case of analogy testing this means being able to detect whether a task is focused on grammar, syntax or a subcategory thereof. For each method we took the top N models for different values of N and evaluated them using our analogy testing framework.

Ranking Score

For each category of analogy test we ranked our word embedding models by their accuracy on that task and assigned each a score. The score for the top model for a category is the number of models being evaluated. The second best word embedding model gets one point less than that, and so on where the nth best model gets 1 point, assuming no ties. These scores are then aggregated across all tasks. We take N word embedding models with the highest ranking scores in order to build the ensemble model.

Sum of Categorical Scores

For each word embedding model we summed their overall performance in each different category and ranked them by average categorical score. This allowed for the possibility of a word embedding model to move up or down in the ranking because of a select few extremely high or low scores, reflecting data that the ranking score does not include.

Maximum Information Selection

The previous two methods are at risk of picking several models that performed very similarly on most tasks. For example if some word embedding model A received the highest accuracy in 70% of test sets, and another word embedding model B received the second best score in the exact same categories, then it is possible that the ensemble model would only consist of A and B. This would not improve the total information leveraged for any task. In effect the ensemble model would likely use the embeddings from A in every situation, not improving overall accuracy at all on the remaining 30% of tasks. Additionally, if some model C exists where C achieves the highest score in one particular test set but the worst in all other test sets it is highly unlikely that it will be included in the ensemble model when using either of the previous two selection methods. This presents a problem because it is obvious that model C has learned some amount of unique, and possibly very useful, information.

To mitigate these problems we introduce Maximum Information Selection. First we create a set of word embedding models whose only element is the top ranked model from either of the previous two methods. Then the remaining word embedding models are ranked by the count of categories in which they achieve a higher score than any model in the set. If this number is nonzero for any model, we take the highest ranked word embeddings by this new methodology and add it to the set of ensemble models. If two models are tied then we use whichever is ranked higher by the method used to select the first model. We repeat this process until either there are no models that can possibly improve the ensemble model, or the desired number of models in the ensemble model has been reached.

This selection process prioritizes increasing the total amount of information used by the ensemble model. It attempts to ensure that any model included in the ensemble model will be useful, and that the maximum amount of language information is represented in the final ensemble model.

3.4.2 Categorical Model Selection

For analogy testing where the categories of each task are known, choosing a word embedding model at run-time is straightforward. Our selection process gives us knowledge of which categories each word embedding model has a higher or lower degree of competency in. Therefore, the ensemble model picks which vectors to use based on the category of the task.

This method allows us to determine the theoretical upper limit of the performance of an ensemble model built with a specific model set. This upper limit is the measure against which any truly dynamic model should be evaluated.

3.5 Analogy Testing

We broke down our test set into its two main categories, grammatical tests and syntactic tests. Each of these two categories was then split further into more subcategories. Under lexical tests there were the subcategories of family, currency, common country capitals, global country capitals, city in state, nationality-adjective and opposites. For grammatical tests we had the categories of comparatives, adjective-to-adverb, present-participle, superlative, plurals, verb plurals and the past tense.

3.5.1 Scoring

For a given question, a model received a point for each question answered correctly and each question is categorized as "answerable" if every word in the test is in the model's vocabulary. Scores are then found across the entire dataset and within each category and subcategory by dividing questions correctly answered by the number of answerable questions. This is therefore a measure of model precision, it is the fraction of all results which were both relevant and correct. It is only possible for the model to return relevant answers as any word from the vocabulary is a relevant answer.

We analyze the rankings of each subcategory as well as the overall variation between the models. This helps to determine which problems are "easier" or "harder" for this kind of model to answer, as well as providing information as to what types of distributional models excel at which particular tasks. We also made use of ranked scoring methods to normalize the effect of certain models being exceptionally good within only one category.

3.6 Implementation

All Java code discussed in this section can be found in the EmoryNLP Github repository, see footnotes for full implementation[1].

3.6.1 Arbitrary and Dependency Contexts

In order to create our new models we expanded on the existing code in the EmoryNLP project. The Word2Vec[2] and Syntactic Word2Vec[3] classes provided the basis for our work. Among our changes the most important was the implementation of context selection based on some input parameter. We replaced the original context selection routine with our more generalized version (see Algorithm 1). Each individual context extraction method was created with the use of the NLPNode object which maintains information regarding dependency and semantic arcs.

We implemented the same changes in the Python implementation of Word2Vec found in the Gensim package as well as adding functionality to utilize dependency and semantic information. However, we found that this method was significantly slower than the Java version. For this reason we used the EmoryNLP implementation for all of our experiments.

[1] www.github.com/emorynlp/text_analysis

[2] github.com/emorynlp/text_analysis/tree/master/src/main/java/edu/emory/mathcs /nlp/vsm/Word2Vec.java

[3] github.com/emorynlp/text_analysis/tree/master/src/main/java/edu/emory/mathcs /nlp/vsm/SyntacticWord2Vec.java

Algorithm 1 Context Selection Procedures

```
procedure ADJACENT
    for all $word$ in $corpus$ do
        $index \leftarrow corpus.getIndex(word)$
        $i \leftarrow -\text{WindowSize}$
        while $i \leq index + \text{WindowSize}$ do
            $train(word, corpus.getWordAtIndex(i)$
            $i \leftarrow i + 1$
        $context \leftarrow wordNode.getContext(contextType)$
        for all $contextWord$ in $context$ do
            $train(word, contextWord)$

procedure ARBITRARY
    for all $word$ in $corpus$ do
        $wordNode \leftarrow NLPNode(word)$
        $context \leftarrow wordNode.getContext(contextType)$
        for all $contextWord$ in $context$ do
            $train(word, contextWord)$
```

3.6.2 Analogy Testing Framework

A single analogy test is performed by extracting the relationship between two words a and a' and then applying it to some word b. The word b is chosen such that there exists some word b' that shares the same relationship with b that a' does with a.

See Algorithm 2 for details. The time complexity of an analogy test is approximately:

$$O(|\vec{x}| \times V)$$

where $\vec{x}$ is an arbitrary vector and V is the size of the vocabulary. This means that for an entire test set, the complexity of evaluating a single model is

$$O(|\vec{x}| \times V \times T)$$

where T is the number of tests.

Generally we have $|\vec{x}|$ on the order of 100, V of approximately 1,531,163 and a test set composed of 19,544 tests. However, running a single test set is highly parallelizable. By implementing a multithreaded approach we decreased the impact of T by a factor of 48[4][5].

The score for an individual model is calculated as *correct/total possible* where *total possible* is the number of tests where each word used is within the model's vocabulary. The single-threaded approach to analogy testing across an entire test set can be found in Algorithm 3.

[4]Using 48 cores with a single thread per core gave us a total testing time of under 20 minutes.

[5]github.com/emorynlp/text_analysis/tree/master/src/main/java/edu/emory/mathcs /nlp/vsm/evaluate/AnalogyTestMulti.java

Algorithm 2 Single Analogy Tests

```
1: procedure ANALOGY TEST(a, b, c, gold, wordVectors)
2:     $d \leftarrow$ SUBTRACTVECTORS(B,A)
3:     $d \leftarrow$ ADDVECTORS(D,C)
4:     $bestMatch \leftarrow$ NULL
5:     $maxSimilarity \leftarrow$ MIN_VALUE
6:     for all $wordVector$ IN $wordVectors$ do
7:         if $max <$ COSINESIM$(d, wordVector)$ then
8:             $max \leftarrow$ COSINESIM$(d, wordVector)$
9:             $bestMatch \leftarrow wordVector$
10:    if $d = gold$ then
11:        return True
12:    else
13:        return False
1: procedure COSINESIM(x,y)
2:     $dotProduct \leftarrow 0$
3:     $xFactor \leftarrow 0$
4:     $yFactor \leftarrow 0$
5:     $i \leftarrow 0$
6:     while $i < x$.LENGTH do
7:         $dotProduct \leftarrow dotProduct + x[i] * y[i]$
8:         $xFactor \leftarrow xFactor + x[i] * x[i]$
9:         $yFactor \leftarrow yFactor + y[i] * y[i]$
10:        $i \leftarrow i + 1$
11:    return $dotProduct/($SQROOT$(xFactor) *$ SQROOT$(yFactor))$
```

Algorithm 3 Analogy Testing Framework

```
1: procedure ALL ANALOGY TESTS(TestPairs,wordVectors)
2:     $score \leftarrow 0$
3:     for all $A$, $B$, $C$, $Gold$ in $TestPairs$ do
4:         $testDidPass \leftarrow$ ANALOGY TEST($A, B, C, Gold, wordVectors$)
5:         if $testDidPass$ then
6:             $score \leftarrow score + 1$
7:     return $score/TestPairs$.LENGTH
```

3.6.3 Ensemble Models

For this research we implemented the ensemble model as a subclass of the analogy testing framework in order to integrate the two modules as much as possible. Each word embedding model is loaded along with its rank in each linguistic subcategory. During analogy testing the category of each test is compared to the list of model ranks and selects the best possible model. A dynamic ensemble implementation would replace the model selection process by a classifier meant to determine which category the task is a member of[6]. In Algorithm 4 we lay out the way in which we change our analogy testing framework to utilize an ensemble model. The high-level changes to the algorithm are kept simple to maintain flexibility in attempting different strategies for dynamic model selection. Our current model selection model is essentially picking a string from a list and so was left out as trivial.

Algorithm 4 Ensemble Analogy Tests

```
1: procedure ENSEMBLE ANALOGY TESTS(TestPairs, embeddingModels)
2:     $score \leftarrow 0$
3:     for all $A$, $B$, $C$, $Gold$ in $TestPairs$ do
4:         $wordVectors \leftarrow embeddingModels$.SELECTMODEL($a, b$)
5:         $testDidPass \leftarrow$ ANALOGY TEST($A, B, C, Gold, wordVectors$)
6:         if $testDidPass$ then
7:             $score \leftarrow score + 1$
8:     return $score/TestPairs$.LENGTH
```

[6]github.com/emorynlp/text_analysis/tree/master/src/main/java/edu/emory/mathcs/nlp/vsm/evaluate/EnsembleAnalogyTest.java

Chapter 4

Experiments

Before any other experiment we built a word embedding model using the EmoryNLP implementation of Word2Vec. Our theory at this point was that there were some particular tasks on which Word2Vec would perform better because of biases in its training. After this we designed and built alternative models rooted in syntactic theory. We intended to show that models built with different linguistic information have unique competencies on different language tasks, eventually composing the trainings of different models together to maximize the total useful information that could be learned at once. This would provide a way to build higher quality embeddings while also demonstrating that meaning learned by a distributional semantic model can be enhanced by optimizing the type of syntactic information extracted in training.

4.1 EmoryNLP Word2Vec

Table 4.1: Word2Vec Iteration Scores

Iterations	**Grammatical**	**Lexical**	**Overall Score**
1	0.2561	0.0552	0.2080
2	0.1923	0.0311	0.1535
3	0.2367	0.0541	0.1907
4	0.2561	0.0552	0.2080
5	0.6289	0.0047	0.2686

We first analyzed Word2Vec's training process. Table 4.1 shows that by the end of its 5th training iteration Word2Vec extracts a huge amount of grammatical information from its training corpus. Interestingly, it seems like improvement in linguistic competency comes at the cost of losing most lexical information learned during this process.

Figure 4.1 shows the lexical competency of Word2Vec almost completely disappearing just as the model's performance in grammatical categories surges. It is notable that performance across both categories generally trend together until the 5th iteration where they abruptly diverge.

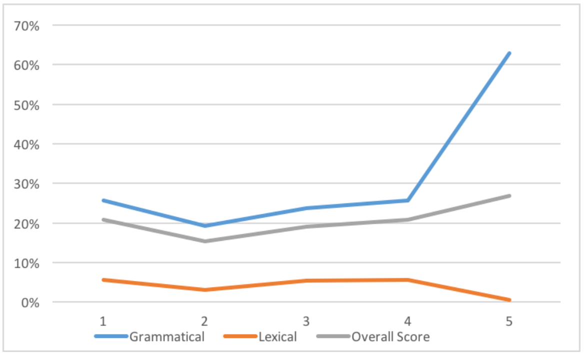


Figure 4.1: Word2Vec Lexical and Grammatical Performance Trends Over Training Iterations

While the decrease of lexical competency seems like a substantial drawback to training Word2Vec until the 5th iteration, the overall improvement driven by grammatical information actually substantially increases the model's overall quality.

Our goal at this point was to determine whether the traditional method of training Word2Vec led to particular competencies or deficiencies in the model. The answer to this was a resounding "yes". Table 4.1 shows that Word2Vec performs very well on grammatical tasks but poorly on lexical oens.

Table 4.2 breaks down the Lexical and Grammatical tests into their component analogy

Table 4.2: EmoryNLP Word2Vec Categorical Scores

Analogy Test Set	**Category**	**Score**
plural-verbs	Grammatical	0.94023
plural	Grammatical	0.74249
past-tense	Grammatical	0.73269
superlative	Grammatical	0.69430
present-participle	Grammatical	0.65625
comparative	Grammatical	0.58183
nationality-adjective	Lexical	0.01313
family	Lexical	0.00988
capital-common-countries	Lexical	0.00791
capital-world	Lexical	0.00442
opposite	Lexical	0.00123
city-in-state	Lexical	0.00081
currency	Lexical	0.00000
adjective-to-adverb	Grammatical	0.00000

test sets. While we had already shown that Word2Vec does not very effectively learn lexical information, it is interesting to see that there is not a single exception to this rule. There is so little lexical information extracted by this model that only a single non-grammatical test set received over 1% precision across all of its tests.

The particular competencies of the EmoryNLP Word2Vec model are made clear by its precision score on the $adjective-to-adverb$, the category it learned the least material about, acheiving a 0% score. This is telling because $adjective-to-adverb$ is the only grammatical category in which Word2Vec could not correctly answer at least 50% of all questions. The tests that comprise $adjective-to-adverb$, as shown in Table 2.1, concern themselves with relations of the form *Amazing is to Amazingly* or *Slow is to Slowly.*

Extracting and applying the relationship between these words transforms the part-of-speech of the target, a unique trait among all other grammatical tests. $adjective-to-adverb$ is the only grammatical test set that requires knowledge of derivational morphemes, every single other test set is based upon inflectional morphemes. This narrows the competency of the Word2Vec model to a much more concrete subset of linguistic information, it excels at learning information regarding inflectional morphemes.

4.2 Lexical Evaluation

The adjacency-window approach of Word2Vec is seemingly only capable of learning information about inflectional morphemes and grammar. The case for this claim is bolstered by the results of our basic and composite syntactic word embedding models.

Table 4.3: Lexical Tests

Model	**Score**
dep2	0.2836
dep2h	0.2832
AllSiblings	0.2551
sib1dep1	0.2189
sib1dep2	0.1890
sib2dep1	0.1841
srl1	0.1731
dep1	0.1496
sib2	0.1486
sib1	0.0781
w2v	0.0047

Table 4.3 shows that every single alternative context extraction method we attempted was at least an order of magnitude more effective than that used by Word2Vec in terms of lexical information. It provides a high-level look at each model's overall competence across a variety of specific tasks. Based on this data, the *dep*2 approach of training on all of the dependents of each word and on their dependents is the most generally useful word embedding model for lexical tasks. It also seems that training on the head of a node, at least in this case, can decrease the quality of distributional word representations. It should be noted that the top

few models all performed very similarly, even though they were occasionally trained with disjoint training methods. That is, the *dep*2 model and *AllSiblings* model seemed to have extracted almost the same amount of language information from our corpus, even though *dep*2 never trained any word on its sibling, and *AllSiblings* does not utilize a single word's dependents.

The information from Table 4.4 includes the competencies of each model across each lexical task category. We can quickly ascertain from this data a few important facts. First, no single context generation approach was uniquely more adept at extracting and learning lexical information. Second, in every lexical category the most information was learned by either a composite model or an extended version of a basic model. Perhaps even more important is the fact that it was only the simplest composite models that were successful, more complicated methods like *sib*1*dep*2 not only did worse than the sum of their predecessor methods, but often even performed poorly compared to composite models that used a subset of their training contexts. Specifically, *sib*1*dep*1 frequently was found to have built higher quality word representations than either *sib*1*dep*2 or *sib*2*dep*1. This is particularly striking in respect ot *sib*2*dep* considering that at most each context will only contain two additional words, and that in generally adding additional siblings to the training context seemed to be helpful in other models such as *sib*2. In the lexical categories of $capital-world$, $opposite$, $currency$, $capital-common-countries$ and $nationality-adjective$ it is found that *sib*2 is a significant improvement over *sib*1.

The decreased quality of the *sib*1*dep*2 and *sib*2*dep*1 models are a clear indication that more training information does not always guarantee an improvement in word embedding quality. In fact even adding context information that seems to generally work well in other composite

Table 4.4: Lexical Score Breakdown

gender	**Score**
sib1dep1	0.2154
AllSiblings	0.2134
sib1	0.1680
dep2	0.1423
dep2h	0.1423
sib2	0.1383
srl1	0.1304
sib2dep1	0.1087
dep1	0.1087
sib1dep2	0.0731
w2v	0.0099

capital-world	**Score**
dep2	0.3722
dep2h	0.3689
AllSiblings	0.3081
sib1dep2	0.2314
sib1dep1	0.2202
srl1	0.2053
sib2dep1	0.2016
dep1	0.1691
sib2	0.1477
sib1	0.0595
w2v	0.0044

opposite	**Score**
AllSiblings	0.0591
sib1dep1	0.0530
dep2h	0.0517
dep1	0.0493
dep2	0.0480
srl1	0.0369
sib2dep1	0.0345
sib1dep2	0.0308
sib2	0.0296
sib1	0.0197
w2v	0.0012

currency	**Score**
AllSiblings	0.0751
sib1dep1	0.0751
sib2	0.0670
sib2dep1	0.0658
sib1dep2	0.0612
sib1	0.0485
dep2	0.0381
dep2h	0.0370
srl1	0.0162
dep1	0.0104
w2v	0.0000

capital-common countries	**Score**
dep2h	0.5632
dep2	0.5257
AllSiblings	0.5099
sib1dep2	0.4802
sib1dep1	0.4466
srl1	0.4328
dep1	0.3972
sib2dep1	0.3755
sib2	0.3123
sib1	0.1897
w2v	0.0079

nationality adjective	**Score**
dep2	0.6648
dep2h	0.6579
sib1dep1	0.6310
AllSiblings	0.6085
sib2dep1	0.5009
sib1dep2	0.4472
sib2	0.4340
srl1	0.4271
dep1	0.3759
sib1	0.2239
w2v	0.0131

city-in-state	**Score**
dep2h	0.0170
dep2	0.0170
sib2dep1	0.0138
AllSiblings	0.0130
sib1dep1	0.0085
dep1	0.0069
sib1	0.0061
srl1	0.0049
sib1dep2	0.0049
sib2	0.0016
w2v	0.0008

models does not necessarily indicate that the language model will improve. These results suggest that there may be a point after which training on additional dependency information will not provide any benefit at all. However we recognize that it is certainly possible that we simply did not train enough complicated models to discover which combinations of linguistic features are complementary.

One reason these models may have done so poorly is that their constituent models could have actually been learning at least some of the same information to start with. That is to say, it is possible that some overlap existed in information between $sib2$ and $dep1$ such that $sib2dep1$ would end up distorting distorting this information through over-training. It's certainly plausible that a sibling node and a dependent node could convey similar information about a node they are both connected to. This could result in the composite model unnecessarily over-training on certain (possibly useless) information and losing quality in the process.

4.3 Grammatical Evaluation

As we saw previously, building contexts from a window of adjacent words in order to perform a training task is an incredibly effective way to extract grammatical relationships between words. The most striking aspect of this fact is that this competency seems exclusive to adjacency-context training. Not a single syntactic context model achieved over 3% precision on any task which involved inflectional morphemes while Word2Vec consistently displayed precision of anywhere from 58% to 94%. The single grammar task in which syntactic models learn more information than the Word2Vec model is $adjective - to - adverb$. Even the $adjective - to - adverb$ test set is an outlier in that it is the single relationship that concerns

Table 4.5: Grammatical Score Breakdown

present-participle	**Score**
w2v	0.6563
srl1	0.0199
sib1dep1	0.0189
dep2h	0.0152
dep1	0.0114
AllSiblings	0.0104
dep2	0.0095
sib1dep2	0.0038
sib2dep1	0.0028
sib2	0.0019
sib1	0.0009

plural-verbs	**Score**
w2v	0.9402
dep2	0.0012
sib1dep2	0.0012
dep1	0.0012
sib1dep1	0.0012
sib2dep1	0.0000
sib1	0.0000
srl1	0.0000
dep2h	0.0000
AllSiblings	0.0000
sib2	0.0000

comparative	**Score**
w2v	0.5818
AllSiblings	0.0120
sib1dep1	0.0075
sib2dep1	0.0053
dep2	0.0038
sib2	0.0030
dep2h	0.0030
sib1dep2	0.0030
dep1	0.0015
srl1	0.0008
sib1	0.0000

past-tense	**Score**
w2v	0.7327
srl1	0.0013
AllSiblings	0.0013
sib1dep1	0.0013
sib1	0.0006
dep1	0.0006
sib2dep1	0.0000
dep2	0.0000
dep2h	0.0000
sib1dep2	0.0000
sib2	0.0000

plural	**Score**
w2v	0.7425
AllSiblings	0.0031
srl1	0.0023
dep2	0.0023
sib1dep1	0.0023
dep2h	0.0015
dep1	0.0015
sib1dep2	0.0008
sib2	0.0008
sib1	0.0000
sib2dep1	0.0000

superlative	**Score**
w2v	0.6943
sib1dep1	0.0223
AllSiblings	0.0187
sib2dep1	0.0134
dep2h	0.0080
dep2	0.0071
srl1	0.0045
sib1dep2	0.0045
dep1	0.0036
sib2	0.0036
sib1	0.0000

adjective-to-adverb	**Score**
srl1	0.0524
sib1dep1	0.0494
dep1	0.0433
dep2h	0.0383
AllSiblings	0.0383
sib1	0.0333
dep2	0.0302
sib2dep1	0.0252
sib2	0.0242
sib1dep2	0.0151
w2v	0.0000

derivations morphemes, decidedly the more lexical of the two categories.

4.3.1 Rank Scoring

Table 4.6: Rank Scores

	Grammatical Rank	**Lexical Rank**	**Total**
sib1dep1	74	63	137
AllSiblings	69	67	136
dep2h	61	64	125
dep2	58	64	122
srl1	65	37	102
sib2dep1	54	45	99
dep1	59	35	94
sib1dep2	50	43	93
w2v	75	12	87
sib2	47	35	82
sib1	45	30	75

After gathering our initial results and ranking them by category we wanted to get a better idea of which models were more generally useful across all categories. We took each model and assigned it points for each category, where the model that ranked highest for a particular category would get a number of points equal to the number of models we were examining, the second highest ranked model would get one point less than that and so forth. Using the analysis shown in Table 4.6 we determined that six of the syntactic models we built performed better on a wider range of tasks than Word2Vec did, while four of our models were generally less useful than Word2Vec. *All siblings*, *sib*1*dep*1, *dep*2, *dep*2*h*, *srl*1 and *dep*1 all received a higher ranked score than Word2Vec did. This analysis indicates that models using the *sib*1*dep*1 and *AllSiblings* context

4.4 Context Analysis

We found that while the previous Word2Vec training approach performs particularly well on inflectional morphemes, every model that leveraged syntactic information to build training contexts outperformed Word2Vec on every lexical task in our inventory. In order to start to determine why these training methods capture such different information we first look at the difference between the contexts of traditional Word2Vec and our new approach.

Table 4.7: Context Statistics

	New Information	**Lexical Score**	**Grammatical Score**	**Language Score**
w2v	0.000	0.005	0.629	0.269
dep2h	0.742	0.283	0.008	0.168
dep2	0.702	0.284	0.007	0.167
allSiblings	0.559	0.255	0.011	0.152
sib1dep1	0.832	0.219	0.013	0.132
sib1dep2	0.751	0.189	0.004	0.111
sib2dep1	0.770	0.184	0.006	0.109
dep1	0.821	0.150	0.008	0.090
dep1	0.821	0.150	0.008	0.090
sib2	0.738	0.149	0.004	0.088
sib1	0.843	0.078	0.004	0.047

For simplicity we'll focus on models that exploit dependency relationships. For each context building approach we counted the total number of words that would be included in that context over the course of a single training iteration. We then determined how many of those words would not have been included if a adjacent context-window approach had been used, assuming a window size of 5.

The reason we did this is that one might suspect that training on words that are not present in Word2Vec's context window would correlate with a word embedding gaining competency in categories where Word2Vec is not able to build a high quality model. This would mean that the improvements we have created are a result of simply choosing words to train on that are further than 5 positions away, not because of the use of linguistic information to build contexts.

However, given the data in Table 4.7 we can see that this is not the case. In fact, it almost seems as if more utilization of new information negatively correlates with lexical model strength based on Figure 4.2. We certainly have too little data to make any formal claim along these lines, but this does tell us that it is not an explicit trade-off, more information outside of the context window does not necessarily increase competency in lexical tasks. The model which used all dependency siblings as context actually only used roughly 56% different information than Word2Vec but was one of the top performing models. This indicates that building word embedding models for lexical tasks is mostly affected by improving the way in which each word is selected, not by the context size itself or the position of the context words within the sentence.

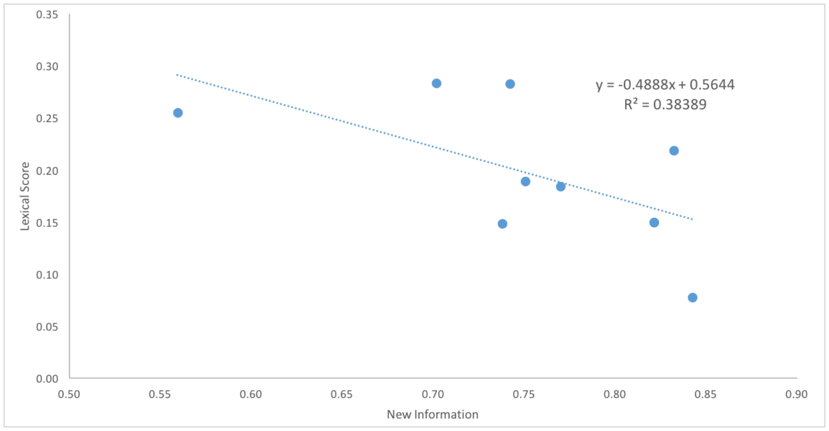


Figure 4.2: Lexical Competency vs. Percentage of Training Words Outside of Word2Vec Context Window

4.5 Ensemble Models

We built our ensemble models using the full set of our distributional representation models. By ranking model performance and selecting ensemble members with three different methods we aimed to dynamically detect the best possible way to answer any given analogy test.

Table 4.8: Ensemble Analogy Scores

Ensemble	Maximum Model Size	Score
maxinfo	7	0.437218584
rank	11	0.437218584
rank	10	0.437218584
rank	9	0.437218584
maxinfo	6	0.437065084
sum	12	0.436927954
sum	10	0.436905614
sum	9	0.436905614
sum	11	0.436905614
maxinfo	5	0.436809251
sum	6	0.436752224
sum	8	0.436752224
sum	7	0.436752224
sum	5	0.436138664
maxinfo	4	0.435478919
sum	4	0.434809285
maxinfo	3	0.431283258
sum	3	0.430974537
maxinfo	2	0.268624642
maxinfo	1	0.268446081
sum	1	0.268432355
rank	5	0.175419683
rank	6	0.175419683
rank	7	0.175419683
rank	8	0.175419683
rank	4	0.175214333
rank	3	0.173879563
rank	2	0.155654808
rank	1	0.132398994

This method was to be used as an alternative to building composite models which were often inefficient in combining competencies from different training methods. We also sought to establish a hypothetical maximum amount of information that could be extracted from word embedding models with different competencies.

Three different ensemble models were tested, Sum of Categorical Scores, Rank Scores and Maximum Information Selection. Each of these was made to select one distributional word representation of the set we had created and then passed through our analogy tests. At the completion of the analogy tests the ensemble would choose another model to add and repeat the process. The algorithm with which each ensemble selects its word representations greatly affects its overall performance and the speed at which each one reaches the theoretical maximum that can be achieved with a finite and known set of word embeddings.

Every method of building sufficiently large ensemble models significantly outperformed any individual model, as shown in Table 4.8. The most efficient model by far was the maximum information selection model, which only used 7 different distributional representations to tie for the maximum analogy test score. In comparison, choosing models by the sum of their categorical scores required 9 models to achieve the same result, and selection by ranking score could not reach this point using 11 different models, the maximum we used to test[1]. Ranking score selection was the least efficient of all of these methods in other regards as well, it required 9 models to achieve a score greater than or equal to EmoryNLP Word2Vec, a threshold which took maximum information selection only 2 models to pass while sum of categorical score selection did the same with 3 models.

It is interesting to note that the 9th model chosen by ranking score selection, the one with which it first achieved a higher accuracy than Word2Vec, had enough unique information

[1]Had we allowed every ensemble to use every model there would have been no difference in performance.

to cause the ensemble to achieve our theoretical maximum score along with several other ensembles. This directly indicates the necessity of optimal model selection in creating an ensemble model, occasionally all of the unique information is in only a few models.

4.5.1 Diminishing Information

A significant trend in the ensemble model results is that after the point at which an ensemble model reaches a certain degree of accuracy it is very difficult to improve it any further. As shown in Figure 4.3, once a model surpasses Word2Vec in terms of accuracy (about 26%) it immediately jumps to approximately 43% and no amount of additional information will improve it beyond an additional percentage point.

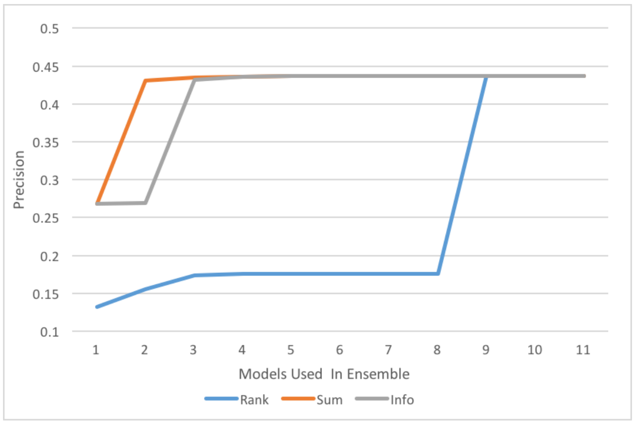


Figure 4.3: Precision by Total Models Included in Ensemble

This implies two realities: 1) much of the important information learned by these distributional representations can be leveraged with a very small, but well constructed, ensemble model and 2) it is unlikely that we can push the ensemble model any further without creating additional models based upon new syntactic information. Based on our results, in fact, it only takes an ensemble model with 2 different distributional representations to achieve accuracy within a percentage point of the maximum that we have observed. Both sum of categorical score selection and maximum information selection achieve this, however for every additional representation that maximum information selects it receives a much more significant accuracy increase than sum of categorical scores receives. After a certain point, even adding four more representation models to sum of categorical scores would not change its performance at all. It is also worth noting that selecting models based on the sum of their categorical accuracies is significantly better than ranked scoring for the first 8 models in this specific circumstance. While this does not necessarily indicate a superior selection process, it definitely indicates that each of these two methods suffers from choosing models that have overlapping competencies.

For these reasons we believe it is clear that maximum information selection is the optimal method to build an ensemble model. In this case composing an ensemble model with only EmoryNLP Word2Vec and *dep2h* provides immense model synergy and captures almost all of the information that our models have extracted.

Chapter 5

Conclusion

In this paper we have explored the distributional hypothesis, but no distributional semantic model currently in use trains each word on each other word it occurs with. This means that, assuming the hypothesis to be correct, we are currently unable to build a distributional language model that learns the total possible amount of meaning from language. However, we can leverage the insight of the hypothesis to build competent but imperfect models.

The difference in linguistic competency between models leads us to believe not only that the meaning of a word can be drawn from its statistical co-occurrences, but also that the degree of meaning extracted from these occurrences is not uniform. That is to say that given some word x there likely exist two words y and z, both of which x occurs with, where there is more information to the meaning of x in its statistical relationship with y than with z.

Having shown that the simple inclusion of new information is not the root of the various competencies of our dependency word embeddings, we feel confident in positing that building training contexts based off of syntactic information allows the model to more efficiently extract language information. Consider the very nature of dependency structure: each pair of tokens connected by an arc has a specific relationship and are therefore related in some way in the language. This is not necessarily the case with adjacent words. By traversing these arcs in building training contexts we are, in a way, overcoming the brute-foce guesswork inherent in using a context window of adjacent terms and choosing to only perform training tasks on pairs of words that have a syntactic relationship.

Our research shows that training using an adjacency context is an incredibly effective method of extracting information about inflectional morphemes from a corpus. It seems that statistical co-occurrence relationships between nearby words conveys a disproportionate amount of information about the foundations of english grammar. Additionally, we find that learning lexical information is very effectively done by leveraging a word's relationship with words below it on a dependency tree as well as the those with which it shares a head.

Practically, this means that there exist different methods of context building that more are more effective at training words together to maximize the total information extracted from their language. By leveraging our analysis framework we were able to identify key competencies in basic word embeddings and construct composite models that captured unique language information. We found that combining the training procedures of basic word embedding models was effective in improving language model competency across specific categories. We presented and implemented an approach that allows for categorical switching of word embeddings depending on current task in order to maximize the total language information used in a specific task. Our best ensemble approach using maximum information selection improves over the original EmoryNLP Word2Vec by 16.9 percentage points, while also achieving the top result for every single analogy category. Our most effective composite model, $dep2$, shows improvement over EmoryNLP Word2Vec of approximately 27.9 percentage points in lexical categories.

5.1 Future Work

Possible future work on this task includes vector space analysis, model training with additional linguistic information, extending the ensemble model framework to automatically detect which distributional semantic representation to use on a given task and a more in-depth linguistic investigation of the competencies of these models.

5.1.1 Additional Models

We have shown that leveraging different linguistic information in the training of distributional models can improve the quality of embeddings in discrete syntactic categories. We built a large model set leveraging dependency and semantic role information in a novel way, while also extending our implementation to ensure that any new training methods would be easy to implement. There remains a massive wealth of linguistic information that can be utilized in training distributional models, much of which may improve language models in ways not yet conceived.

5.1.2 Ensemble Models

We have provided the framework with which to benchmark any ensemble model that performs categorical model switching by extracting syntactic task information. The next step is to devise methods to classify tasks on the fly in order to maximize the relevant model competencies applied to every task. The best case scenario would be developing a reliable method to compose training methods in a way that extracts all of the information learned by single word embedding models, but as we have shown this approach becomes highly unpredictable as more information is leveraged and more contexts are combined. The difficulty of this task makes the usefulness of ensemble models obvious.

5.1.3 Vector Space Analysis

To better understand how to advance distributional semantics we should analyze to a much greater extent the high-dimensional vector spaces of the different models we are building. A stronger grasp of the relationships between different vectors may render some idea as to why certain tasks are failing or why specific training methods provide differing linguistic competencies.

Distance Distortion

It is possible that in local spaces, containing generally similar word vectors, structures will be more similar than in distant spaces. To some degree this appeals to intuition, it doesn't make sense that a cluster of *animal*-related vectors would share perfect high-dimensional organizational similarities with a cluster of *programming language*-related vectors. However, analogy tests and other NLP systems rely on these relationships to be regular and predictable throughout the vector space. Analyzing whether distance between analogy pairs correlates with the probability of successfully completing that particular analogy test would immensely further our understanding of word embedding competency and create a path forward to improve distributional semantic models by manipulating the vector space.

Intra-Cluster Distance Variance

Similar to the idea of relationships between vector pairs distorting over distance, it is also plausible that different semantic clusters might maintain the same local structure but at a different scale. While the vectors within some cluster A might relate to each other in the same way as the vectors within some other cluster B, the magnitude of their vector offsets may be scaled by some multiplicative factor. This would cause the relationships between words in local structures to remain predictable and extensible, but only within similarly structured spaces. An answer to this question could be sufficiently generated by analyzing the variance of vector offsets between analogy pairs that belong to the same syntactic category. If this hypothesis were found to be accurate it would allow for the improvement of word embedding models by normalizing intra-cluster vector offsets, thereby universally improving the usefulness of distributional word representations.

Appendix A

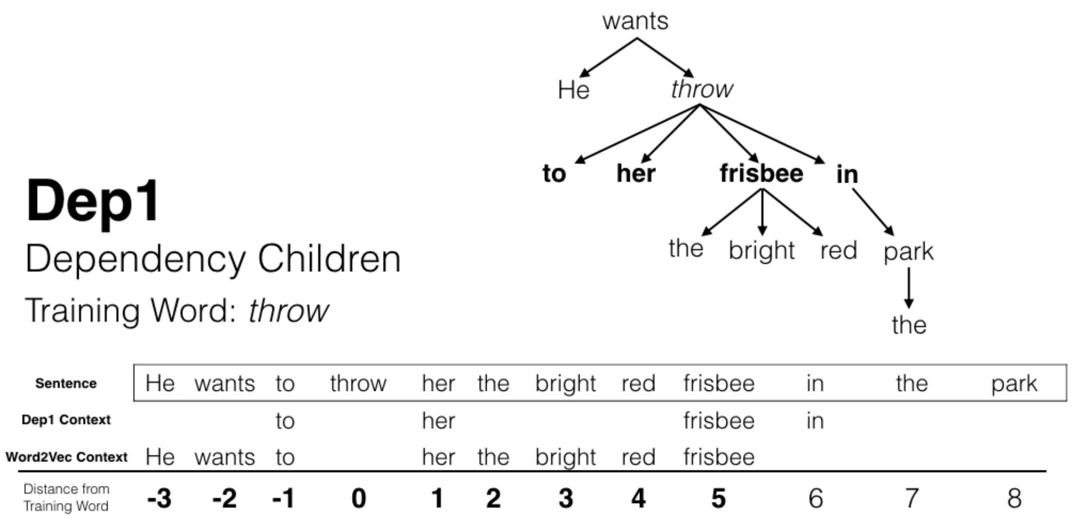
Glossary

Table A.1: Model Abbreviations

Abbreviation	**Description**	**Model Type**
w2v	EmoryNLP Word2Vec	*word embedding*
dep1	Only dependents	*word embedding*
sib1	Nearest right and left sibling	*word embedding*
sib2	Nearest two right and left siblings	*word embedding*
srl1	Semantic Role Head	*word embedding*
allsibs	All nodes that share the current node's head	*composite embedding*
dep2	Dependents and their dependents	*composite embedding*
dep2h	Head node, dependents and their dependents	*composite embedding*
sib1dep1	Dependants and nearest right and left sibling	*composite embedding*
sib1dep2	Dependents, their dependents and nearest right and left sibling	*composite embedding*
sib2dep1	Dependents and nearest two right and left siblings	*composite embedding*
maxinfo	Next model selected has most categorical scores higher than any currently in ensemble	*ensemble*
ranked	Next model selected has the current highest rank score	*ensemble*
sum	Next model selected has the current highest sum score across analogy tasks	*ensemble*

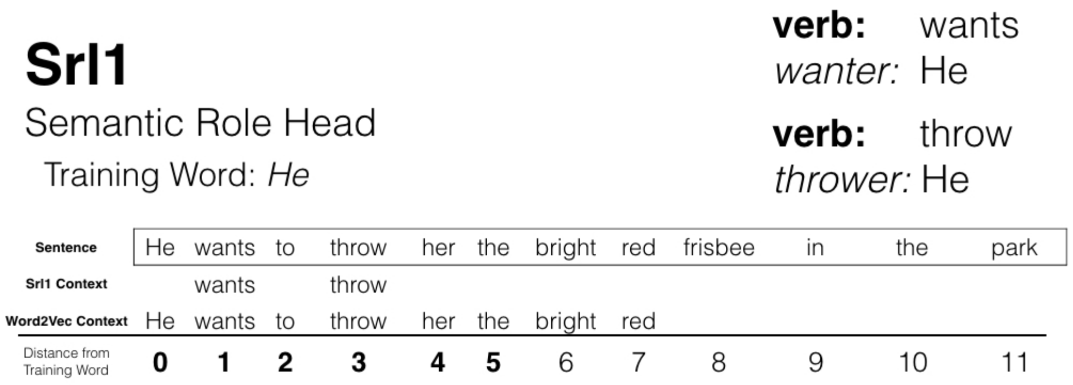
Appendix B

Syntactic Context Comparisons



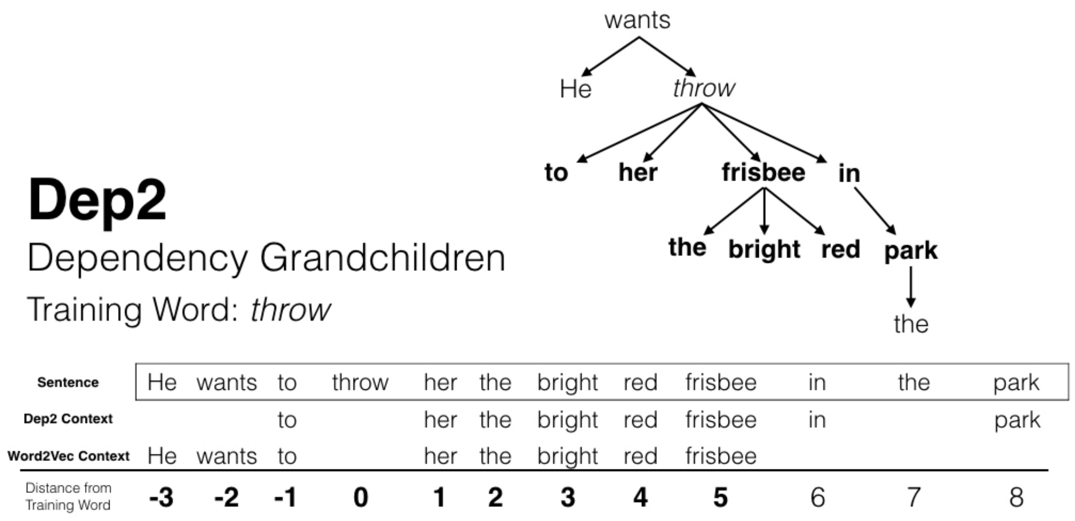
Sentence	He	wants	to	throw	her	the	bright	red	frisbee	in	the	park
Dep1 Context			to		her				frisbee	in		
Word2Vec Context	He	wants	to		her	the	bright	red	frisbee			
Distance from Training Word	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**5**	6	7	8

Figure B.1: Dep1 Model Context Example



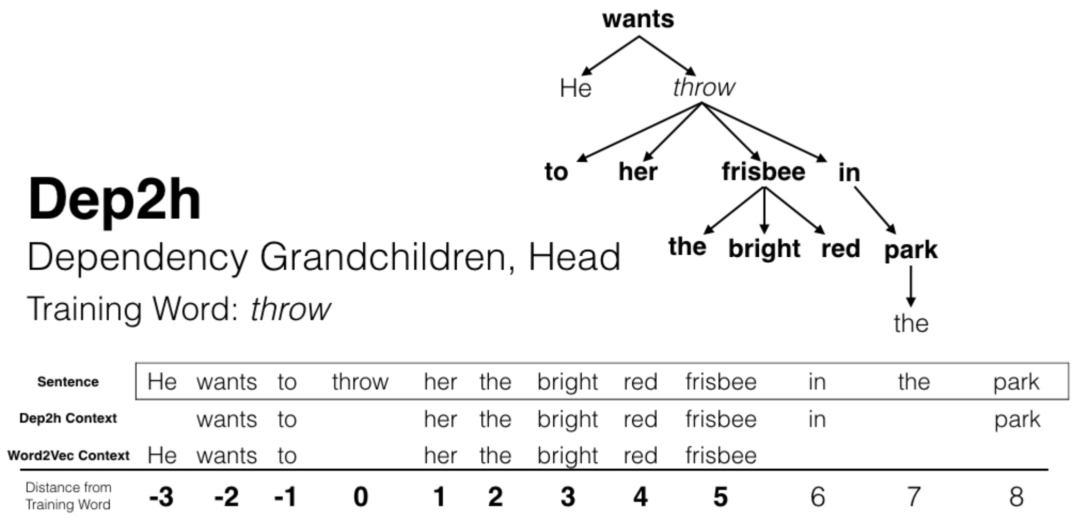
Sentence	He	wants	to	throw	her	the	bright	red	frisbee	in	the	park
Srl1 Context		wants		throw								
Word2Vec Context	He	wants	to	throw	her	the	bright	red				
Distance from Training Word	**0**	**1**	**2**	**3**	**4**	**5**	6	7	8	9	10	11

Figure B.2: Srl1 Model Context Example



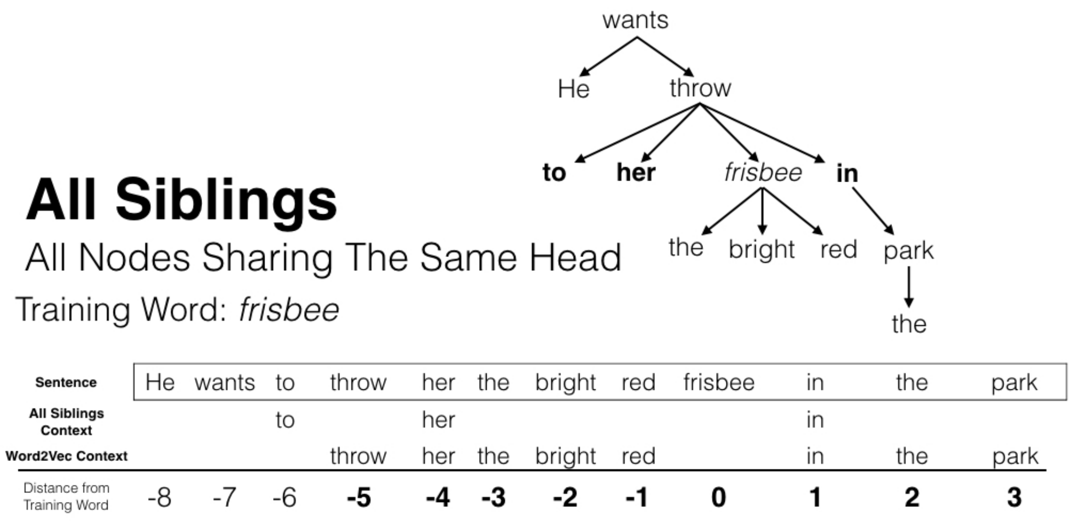
Sentence	He	wants	to	throw	her	the	bright	red	frisbee	in	the	park
Dep2 Context			to		her	the	bright	red	frisbee	in		park
Word2Vec Context	He	wants	to		her	the	bright	red	frisbee			
Distance from Training Word	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**5**	6	7	8

Figure B.3: Dep2 Model Context Example



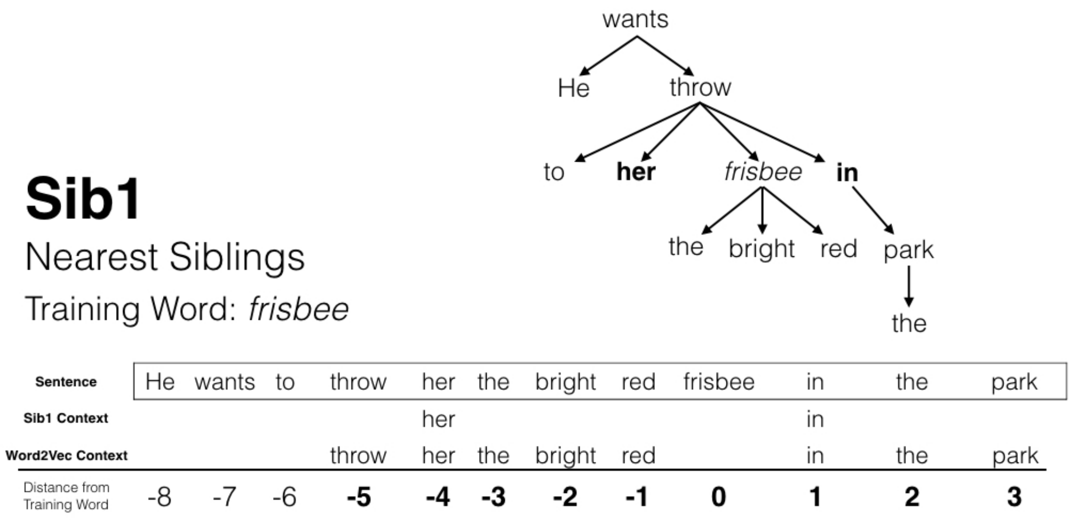
Sentence	He	wants	to	throw	her	the	bright	red	frisbee	in	the	park
Dep2h Context		wants	to		her	the	bright	red	frisbee	in		park
Word2Vec Context	He	wants	to		her	the	bright	red	frisbee			
Distance from Training Word	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**5**	6	7	8

Figure B.4: Dep2h Model Context Example



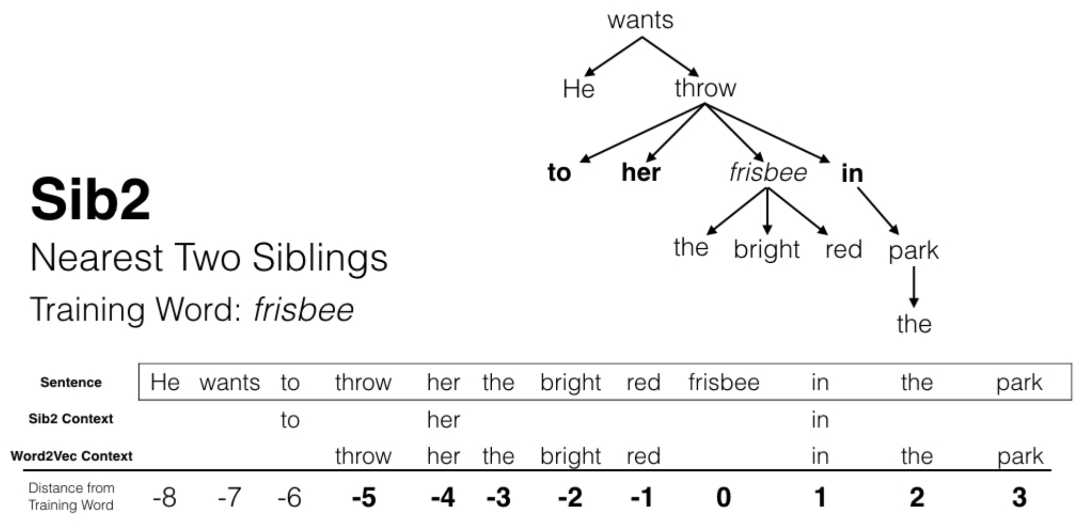
Sentence	He	wants	to	throw	her	the	bright	red	frisbee	in	the	park
All Siblings Context			to		her					in		
Word2Vec Context				throw	her	the	bright	red		in	the	park
Distance from Training Word	-8	-7	-6	**-5**	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**

Figure B.5: All Siblings Model Context Example



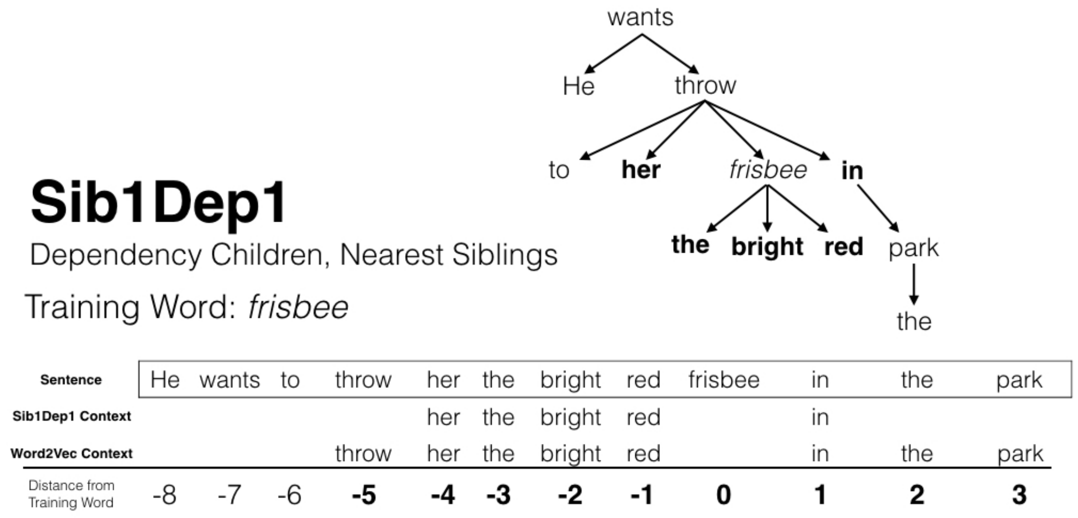
Sentence	He	wants	to	throw	her	the	bright	red	frisbee	in	the	park
Sib1 Context					her					in		
Word2Vec Context				throw	her	the	bright	red		in	the	park
Distance from Training Word	-8	-7	-6	**-5**	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**

Figure B.6: Sib1 Model Context Example



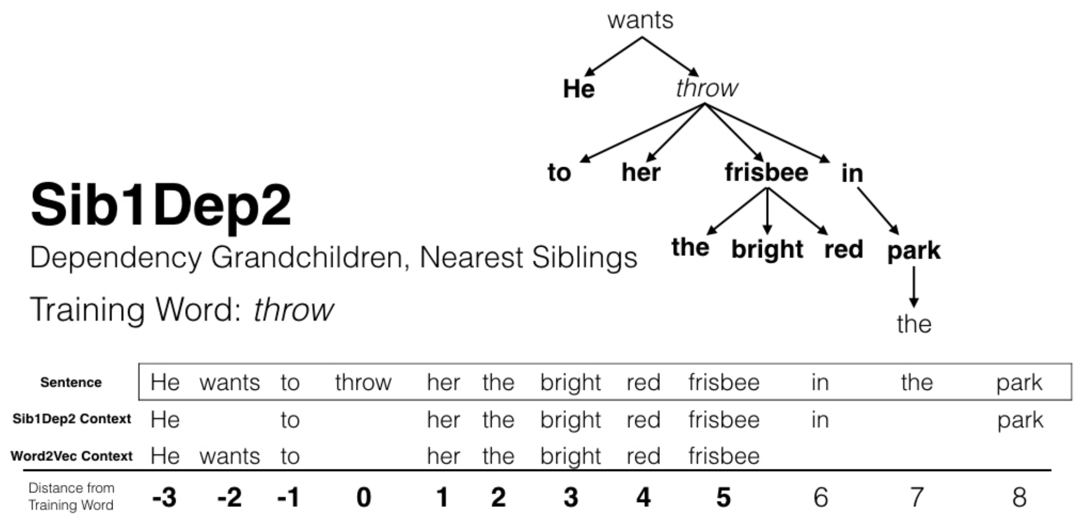
Sentence	He	wants	to	throw	her	the	bright	red	frisbee	in	the	park
Sib2 Context			to		her					in		
Word2Vec Context				throw	her	the	bright	red		in	the	park
Distance from Training Word	-8	-7	-6	**-5**	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**

Figure B.7: Sib2 Model Context Example



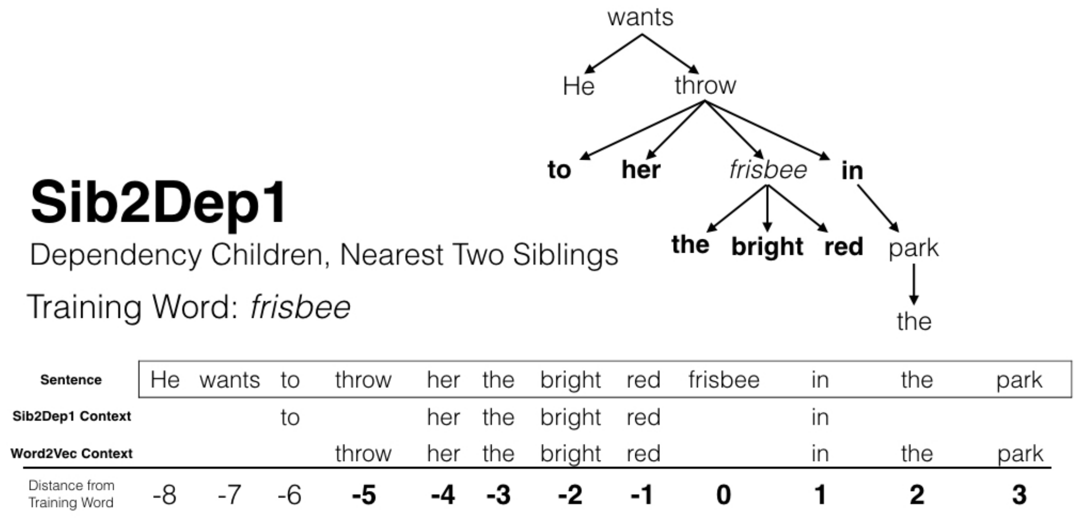
Sentence	He	wants	to	throw	her	the	bright	red	frisbee	in	the	park
Sib1Dep1 Context					her	the	bright	red		in		
Word2Vec Context				throw	her	the	bright	red		in	the	park
Distance from Training Word	-8	-7	-6	**-5**	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**

Figure B.8: Sib1Dep1 Model Context Example



Sentence	He	wants	to	throw	her	the	bright	red	frisbee	in	the	park
Sib1Dep2 Context	He		to		her	the	bright	red	frisbee	in		park
Word2Vec Context	He	wants	to		her	the	bright	red	frisbee			
Distance from Training Word	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**5**	6	7	8

Figure B.9: Sib1Dep2 Model Context Example



Sentence	He	wants	to	throw	her	the	bright	red	frisbee	in	the	park
Sib2Dep1 Context			to		her	the	bright	red		in		
Word2Vec Context				throw	her	the	bright	red		in	the	park
Distance from Training Word	-8	-7	-6	**-5**	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**

Figure B.10: Sib1Dep2 Model Context Example

Bibliography

[1] Bengio, Y., Schwenk, H., Senécal, J.-S., Morin, F., and Gauvain, J.-L. Neural probabilistic language models. In *Innovations in Machine Learning*. Springer, 2006, pp. 137–186.

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

[3] Choi, J. Emorynlp [computer software]. https://github.com/emorynlp, 2014.

[4] Choi, J. Lecture notes for natural language processing (cs 571), sep 2015.

[5] Dumais, S. T. Latent semantic analysis. *Annual review of information science and technology 38*, 1 (2004), 188–230.

[6] Firth, J. R. A synopsis of linguistic theory, 1930-1955. *Blackwell* (1957).

[7] Goldberg, Y., and Levy, O. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR abs/1402.3722* (2014).

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

[9] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[10] Mikolov, T. *Language Modeling for Speech Recognition in Czech.* PhD thesis, Masters thesis, Brno University of Technology, 2007.

[11] Mikolov, T., Kopeckỳ, J., Burget, L., Glembek, O., and Černockỳ, J. H. Neural network based language models for highly inflective languages. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on* (2009), IEEE, pp. 4725–4728.

[12] Mikolov, T., Yih, W.-t., and Zweig, G. Linguistic regularities in continuous space word representations.

[13] Omer Levy, Y. G. Dependency Based Word Embeddings. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Vol. 2, 302-308.

[14] Řehůřek, R., and Sojka, P. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (Valletta, Malta, May 2010), ELRA, pp. 45–50. http://is.muni.cz/publication/884893/en.

[15] Salton, G., Wong, A., and Yang, C.-S. A vector space model for automatic indexing. *Communications of the ACM 18*, 11 (1975), 613–620.

[16] Tomas Mikolov, Kai Chen, G. C., and Dean, J. Efficient estimation of word representations in vector space.

[17] Tomas Mikolov, Ilya Sutskever, K. C. G. C., and Dean, J. Distributed representations of words and phrases and their compositionality.

[18] Turney, P. D., Pantel, P., et al. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research 37*, 1 (2010), 141–188.