Distribution Agreement

In presenting this thesis as a partial fulfillment of the requirements for a degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter now, including display on the World Wide Web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Austin Blodgett April 3, 2015

The Verbiverse: Creating a Verb Space with Comparative Methods of Distributional Semantics

By

Austin James Blodgett

Jinho Choi
Adviser

Department of Mathematics and Computer Science

Jinho Choi

Adviser

Phillip Wolff

Committee Member

Marjorie Pak

Committee Member

2015

The Verbiverse: Creating a Verb Space with Comparative Methods of Distributional Semantics

By

Austin James Blodgett

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

2015

Abstract

The Verbiverse: Creating a Verb Space with Comparative Methods of Distributional Semantics

By Austin James Blodgett

Computational semantics as a field includes many of the unsolved problems of Natural Language Processing. The need for innovation in this field has motivated much research in developing word and language models that better represent meaning and various concepts within semantics. This thesis is concerned specifically with measuring verb similarity and verb clustering, a task within this field. The goal is to develop representations of verbs that can accurately and viably be used to judge semantic similarity between verbs and to group verbs into classes that reflect their relatedness in meaning. Verb clustering – a task of distributing verbs into semantically related classes - has in previous research been shown to have applications in multiple tasks in Natural Language Processing including word sense disambiguation. This thesis will present and compare several methods of automatic acquisition of verb similarity, with a goal of allowing future applications of these methods in NLP tasks and to promote discoveries in how the mechanisms modeled by these methods relate to linguistics.

This paper presents several methods from verb clustering based on Latent Dirichlet Allocation – a probabilistic graphical model commonly used for topic modelling. We model verbs as collections of contextual features derived from latent classes. LDA, which is designed as a model for Bayesian inference of latent thematic categories, fits well to model verb classes based on linguistic context. We demonstrate Recursive LDA, a procedure of executing LDA

iteratively to produce a hierarchical structure of classes. We test several linguistic features from syntax and lexical arguments of verbs with interest in identifying how informative each feature is. We evaluate all of our experiments against human judgments of similarity providing a novel method for evaluating semantic similarity metrics of word models. We test all of our data on a list of 3,000 most common English verbs.

We test our method against Word2Vec, a popular and recently developed word model using skip-gram feature vectors refined by deep learning. The results in this thesis will show that given the right features, our method of using LDA with linguistic features outperforms Word2Vec's data-driven statistical approach when weighed against human judgements.

The Verbiverse: Creating a Verb Space with Comparative Methods of Distributional Semantics

By

Austin James Blodgett

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

2015

Table of Contents

1. Introduction 1
2. Related Work 2
    - 2.1 Constructing Verb Classes 4
    - 2.2 Word2Vec 7
    - 2.3 Comparison to Our Approach 8
3. Linguistic Context and Meaning 9
    - 3.1 Meaning Acquisition Process 10
    - 3.2 Syntax 11
    - 3.3 Lexical Arguments 14
4. Modelling Lexical Semantics 15
    - 4.1 The Instance Space 16
    - 4.2 Understanding Meaning 19
    - 4.3 Properties of Words, Senses, and Categories 21
    - 4.4 Context to Meaning 24
5. Latent Dirichlet Allocation (Beyond Topic Modelling) 25
    - 5.1 A Probabilistic Graphical Model of Documents and Topics 26
    - 5.2 LDA with Verbs and Context 31
    - 5.3 Recursive LDA 32
6. Methodology 34
    - 6.1 Corpus 35
    - 6.2 List of Verbs 35

6.3 Parsing Approaches 36

6.4 Features 39

7. Experiments & Results 40

7.1 List of Experiments 40

7.2 Triad Evaluation Task 42

7.3 Results 44

8. Conclusions 46

9. Future Work 47

Bibliography 48

Appendix A (List of 3,000 Verbs) 54

Appendix B (Verb Space Images) 62

List of Figures & Tables

Figure 1. Gave Constituency Tree (Stanford Parser) 12

Figure 2. Gave Dependency Tree 13

Figure 3. Instance Space 16

Figure 4. LDA Graphical Model 27

Figure 5. Constituency tree (Stanford Parser) 37

Figure 6. Dependency Tree 43

Figure 7. Triad Task 45

Figure 8. Comparing Constituency Features 46

Figure 9. Comparing Dependency Features 55

Figure 10. Verb Space 62

Figure 11. Verb Classes from Recursive LDA 63

Figure 12. Think/Believe Verbs 64

Figure 13. Manner-of-Motion Verbs 64

Figure 14. Other Motion Verbs 64

Figure 15. Mental-State Verbs 65

Table 1. Features 39

Table 2. Evaluation Results 44

1 Introduction

The Verbiverse is a research endeavor with interest in three concepts that are central to the idea of language – verbs, meaning, and context. The Verbiverse is a project with the aim of computationally modelling verb meaning based on features of context and using these models to calculate verb similarities, verb classes, and a relational verb space. A core principle of this project is that there is a strong relationship between word meaning and linguistic context. The goal is to develop novel computational methods for measuring the semantic similarity between verbs. With these similarities, we can identify verbs that are close together in meaning, we can cluster semantically related verbs together into verb classes, and we can create a space of verbs based on semantic similarities between them.

This paper will present several methods of modelling semantic similarity between verbs. We present two methods based on Latent Dirichlet Allocation, an algorithm commonly used for topic modelling (Blei et al., 2003), and we compare results with several other methods including Word2Vec, a word model by Google (Mikolov, Chen, Corrado, & Dean, 2013). We will test our methods with a number of different features of linguistic context with special interest in the advantages of using syntax versus lexical arguments. We test features from two different parsing approaches – constituency parsing and dependency parsing – to test advantages of each. We also present a novel method of evaluating verb similarity called a triad evaluation task. All of the experiments presented in this paper will be compared to human judgements taken from our evaluation task. All of our experiments are used to calculate similarities for a set of 3,000 verbs in English.

Verb clustering based on semantic similarity has been shown in other research projects to be useful for a wide range of Natural Language Processing tasks. These include semantic parsing, word sense disambiguation, and semantic role labelling (Shi & Mihalcea, 2005; Dang, 2004; Swier & Stevenson, 2004; Zapirain et al., 2008). In addition to future applications of this research, we hope this paper and future work of the Verbiverse project will motivate further linguistic research of language acquisition and relationships between word context and meaning. The results in this paper will show that our approach of LDA as a probabilistic representation of verb context outperforms Word2Vec – the leading big-data driven approach to word modelling – when evaluated against human judgement.

Section (2) will provide background and discuss previous research related to this project including other verb classification algorithms. Section (3) will discuss linguistic context and its relationship to meaning. Section (4) presents an abstract model for a formal theoretical approach to lexical semantics. This model will be useful for building an understanding of lexical semantics and how it may be approached from a computational perspective. Section (5) will give an introduction to Latent Dirichlet Allocation (LDA) and our extension of it. Section (6) will discuss details of the data used in this project including corpora and features. Section (7) will present each of our experiments and results. Section (8) will give conclusions. Section (9) will discuss future work.

2 Related Work

This project is part of a broad field of **computational lexical semantics** and more specifically **distributional semantics**. Computational lexical semantics has the aim of developing

computational models of word meaning. This can include manually defining semantic properties of words and semantic relationships between words (synonymy, hyponymy, etc.) and it can also include any data representation of a word that is intended to be representative of meaning. Distributional semantics is a subfield with the aim of modelling meaning using features of context. **Context** in general is any information accompanying an utterance that might be used to infer meaning. This can include physical surroundings, world knowledge, prosody, etc. **Linguistic context** of a word or utterance refers specifically to surrounding words and any relationships (syntactic or semantic) with those words.

A fundamental foundation of distributional semantics and to some fields of study within language acquisition is that there is a strong relationship between a word's linguistic context and its meaning. This is expressed in the **distributional hypothesis**, attributed to linguists J. R. Firth and Zellig Harris, which states that words with similar meaning will also have similar context distributions (Harris, 1954). Firth states:

> You shall know a word by the company it keeps. (1957)

Distributional semantics is traced back to the distributional hypothesis, that it is possible to observe the frequencies of a word's co-occurrences with different context features and compare this distribution of context features to that of another word in order to compare their meanings. Related ideas have been researched in the study of language acquisition. Specifically, the syntactic context of a word has been shown to be useful in the acquisition of word meaning in language acquisition scenarios. This concept called **syntactic bootstrapping** was first researched by Brown (1957) and it has since been demonstrated that infants can use syntactic properties such as verb transitivity (Naigles 1990) and noun count-ness and mass-ness

(Chierchia, 1994) to interpret the meanings of new words. Linguist Beth Levin goes further to identify English **verb classes** - categories of verbs that share syntactic properties and also share aspects of meaning (1993).

2.1 Constructing Verb Classes

Verb clustering (or verb classification) is the task of grouping verbs together into semantically meaningful classes. Often this is done to reflect previously proposed classes like Levin's verb classes (1993). Verb clustering is a task within distributional semantics. This section will discuss various methods of verb clustering that have been used, features that have been used, and applications.

Verb classes are created to group verbs together into meaningful semantic groups. In other words, the verbs in a verb class should all have in common some aspect of meaning. So once a set of verb classes is identified, the verb class labels and other data can be used as features that are closely related to meaning. These features can be applied in many NLP tasks that rely on verb meaning. Verb clustering has been shown to be useful for a number of NLP tasks including semantic parsing, word sense disambiguation, and semantic role labelling (Shi & Mihalcea, 2005; Dang, 2004; Swier & Stevenson, 2004; Zapirain et al., 2008).

These applications have motivated the use of manually annotated verb classes and relationships as well as generative algorithms for verb clustering. The concept of verb classes is often traced back to the linguist Beth Levin (1993) who outlined 49 groups of verbs (and also some subgroups) she observed as having unique and identifiable syntactic and semantic traits. There are also verb lexicons such as WordNet and VerbNet that have been manually labelled to

characterize verbs into semantic groups and identify semantic relationships between verbs (Miller, 1995; Kipper-Schuler, 2005). WordNet and VerbNet are designed to be accessible for NLP tasks and used as features. Some verb clustering project set out to match and build on verb classes of Levin, WorNet, and VerbNet (Kipper et al., 2006; Kipper et al., 2008). Supervised verb classification algorithms use information from sources like WordNet, VerbNet and others like FrameNet, and PropBank as features and use them to train a classifier such as support vector machines or maximum entropy (Merlo & Stevenson, 2001; Sun et al., 2008; Li & Brew, 2008). Some methods combine information from multiple sources (Schulte im Walde, 2006). Semi-supervised algorithms combine supervised and unsupervised approaches using labels from these sources as pivots to generate more features (Stevenson & Joanis, 2003).

Models of verb semantics and distributional semantics in general have been developed with many different mathematical perspectives. Feature vector models rely on principles of linear algebra and treat a verb as a vector in multi-dimensional space (Joanis et al., 2008). A verb with F features is treated as a vector in R^F with dimensionality equal to the number of features. Verbs can be related to each other using various distance metrics such as cosine similarity, Euclidean distance, or Pearson coefficient. The values in each feature vector can be counts of co-occurrence or pointwise mutual information (pmi) which measures statistical dependency between two words or features (Zhou et al., 2011).

$$PMI(x, y) = log \frac{P(x, y)}{P(x)P(y)}$$

Feature vector spaces used as models of distributional semantics are also often called **word spaces**. Once a word space has been calculated, various clustering algorithms (k-means, brown clustering, etc.) can be used to identify coherent groups within the space that are

intended to correspond with verb classes. Since the dimensionality of these spaces is equal to the number of features, often high-dimensionality clustering algorithms are used (Sun & Korhonen, 2009). Linear models of semantics have some natural benefits. For example, they can easily and intuitively be projected into Euclidean space. Vectors can also be added to other vectors or subtracted from other vectors for various kinds of analysis. Section (2.2) will give one example of this for Word2Vec.

Probabilistic models of verb clustering rely on probabilistic properties of context features for different verbs rather than vectors and linear metrics of distance (Reichart & Korhonen, 2013; Merlo & Stevenson, 2001; Vlachos et al., 2009; Dinu & Lapata 2010). Probabilistic models of verb clustering, just like linear models, can produce a space of verbs with distances or similarities between them. Information theory provides several metrics for distributional similarity including Kullback-Leibler divergence.

$$D_{KL}(P||Q) = \sum_i P(i) ln \frac{P(i)}{Q(i)}$$

Other metrics like cosine similarity, Pearson correlation, and Jaccard similarity work as well. A graphical model is the general term for a network of inter-dependent probabilistic variables used to infer latent (non-observed) variables from observed ones. A graphical model can be relatively simple or extremely complex. Latent Dirichlet Allocation (Blei et al., 2003) is one example of a probabilistic graphical model the design for which will be outlined in section (5).

Graph-based verb clustering focusses on defining relationships between verbs and treating the entire structure of verbs as a graph (Sun and Korhonen, 2011; Matsuo et al., 2006).

Once this structure is in place, groups and other properties can be defined in terms of various properties of graphs.

Much previous work in verb clustering has supported the use of syntactic features and acquired subcategorization frames (Sun et al., 2008). Other work has supported the use of lexical features like selectional preference (Sun & Korhonen, 2009) as well as adjunct information (Sun et al., 2008; Joanis et al., 2008), arbitrary word co-occurrence within some window (Li and Brew, 2008), and even tense (Li and Brew, 2008).

2.2 Word2Vec

One research development by a Google project is a program called Word2Vec (Mikolov, Chen, Corrado, & Dean, 2013). Word2Vec can be considered an innovation in distributional semantics in that its purpose is to produce a vector of informative features for each word. Word2Vec is a linear approach to word representation that utilizes deep learning to process massive amounts of non-structured word co-occurrence data into a concise useful feature vector. Word2Vec gets context information from skip-grams - frequencies of word co-occurrences that are allowed to “skip” words in the middle strictly based on words as they appear in linear order. This produces a massive amount of noisy data, but taken from a source with billions of words, deep learning can process this data to produce informative features. Word2Vec feature vectors have been shown to have interesting properties corresponding to aspects of word meaning. For example, vectors for words can be added or subtracted to approximate other words. It’s been shown that the vectors for KING minus a vector for MAN

plus the vector for WOMAN gives a vector very close to the vector for QUEEN (Mikolov, Yih, & Zweig, 2013).

KING – MAN + WOMAN = QUEEN

This is a result of the fact that the features in these feature vectors represent aspects of the contexts of those words. KING and QUEEN share most aspects of context but they differ in the same way that MAN and WOMAN differ. Since the data is modelled linearly, these subsections of context features can be removed or added with simple arithmetic like that above. The appeal of Word2Vec and its approach big data refined with deep learning is that it requires very little understanding of the features being used – in other words the mechanisms of how language works – in order to function. The limits and usefulness of this approach still needs to be thoroughly tested.

2.3 Comparison to Our Approach

This paper presents a probabilistic model of verbs using Latent Dirichlet Allocation (LDA) to identify latent verb categories based on contextual features of syntactic structure and lexical arguments. We calculate similarities between verbs using each verb's distribution across a set of topics and Pearson correlation r between distributions (Pearson performed best experimentally). Pearson correlation for a sample is defined as follows:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

(where $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$)

We will build on previous research into the best linguistic features to use in verb clustering. This paper will argue that our method of verb similarity using LDA intuitively and accurately fits with the structure of verbs, verb classes, and verb meaning. This structure will be described more thoroughly in sections (4) and (5.2). The results from this paper will show that our method of verb clustering with the right features of linguistic context surpasses Word2Vec's approach of data-driven statistical word modelling when weighed against human data. This paper also presents a new method of evaluating verb similarities using human judgment that may be used in future research in this field.

3 Linguistic Context & Meaning

This section is meant to build an understanding of the relationship between context and meaning. This section will build on and explore the idea presented in the distributional hypothesis that words with similar meanings also have similar context distributions. Section (3.1) will present the idea of a meaning acquisition process which will demonstrate the importance of this connection between context and meaning and lay out some of the problems that distributional semantics has the aim of addressing. The rest of this section will discuss specific features of linguistic context and how they might inform aspects of meaning both in a computational model and also theoretically as a process of language acquisition. There are several types of linguistic context discussed in this section for overview. The work in this paper focuses on syntax and lexical arguments.

3.1 Meaning Acquisition Process

The relationship between meaning and context is closely related to a concept that I will call a **meaning acquisition process**. Imagine that you are hearing or reading a sentence in which there is a single word you have never encountered before. Perhaps for example this sentence:

(1) John *treibened* his car from New York to Los Angeles.

Given this sentence by itself, without any previous knowledge of the word *treibened*, the word seems to include some meaning related to "motion." The word *treibened* has surrounding words that seem to provide information of possible meanings of this word. You might replace *treibened* with words like *drove*, *moved,* or *sent*. Notice that words with this related aspect of meaning seem most plausible in this position.

This scenario of context-based word learning represents a major part of both language acquisition and word learning in general. I will use the term meaning acquisition process to refer to this process of inferring the meaning of some word or utterance from context. The word **context** can refer to linguistic context – the context of surrounding words or sentences – or extra-linguistic context such as prosody, body language, and physical context. Most word learning scenarios rely on multiple different aspects of context. A toddler learning the word *clock* will usually rely on physical context (a clock being physically present), body language (a parent pointing at the clock while saying *clock*), and possibly any other information available to

them. Acquisition of the word *clock* presents a very simple case of a meaning acquisition process. But some words, especially words with abstract meanings, require more complex mechanisms of acquisition.

Lexical Semantics has the aim of bettering our understanding of the structure of meaning (how meaning is represented in the brain or how we might represent it in a computer), the compositionality of meaning (how different aspects of meaning come together and are related to each other), and the acquisition of meaning. The meaning acquisition process is thus a core part of lexical semantics as well as lexical semantics. Distributional semantics is focused on the use of linguistic context to represent meaning, and specifically how this process might be modelled computationally. So distributional semantics can be thought of as an attempt to model the meaning acquisition process using a large set of data. There is a great incentive for us to further understand particular aspects of context and the roles they play in the acquisition of meaning. A feature of context that proves to be informative of meaning may give insight into mechanisms of language acquisition and provide a resource for better models of language in Natural Language Processing tasks.
(*treibened* is taken from the German word *treiben* 'to drive' with the English suffix *–ed* mercilessly attached).

3.2 Syntax

For a given verb, there are specific syntactic structures associated with it. Simple examples include transitive and intransitive verbs. Another common structure used in verbs such as *email*, *write*, *send*, and *return* can be seen in the following figures:

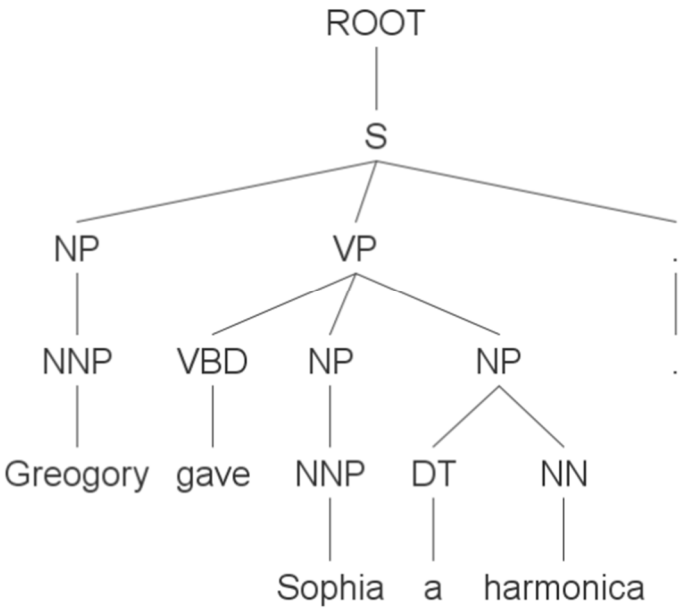

Figure 1. *Gave* Constituency Tree

In the constituency tree above, the verb *give* takes three arguments – a subject, a direct object, and an indirect object. This is a common structure for certain English verbs, and when these verbs are used with the above structure, they all connote a particular aspect of meaning of "transfer from one person to another." Here is a dependency tree representing the same sentence:

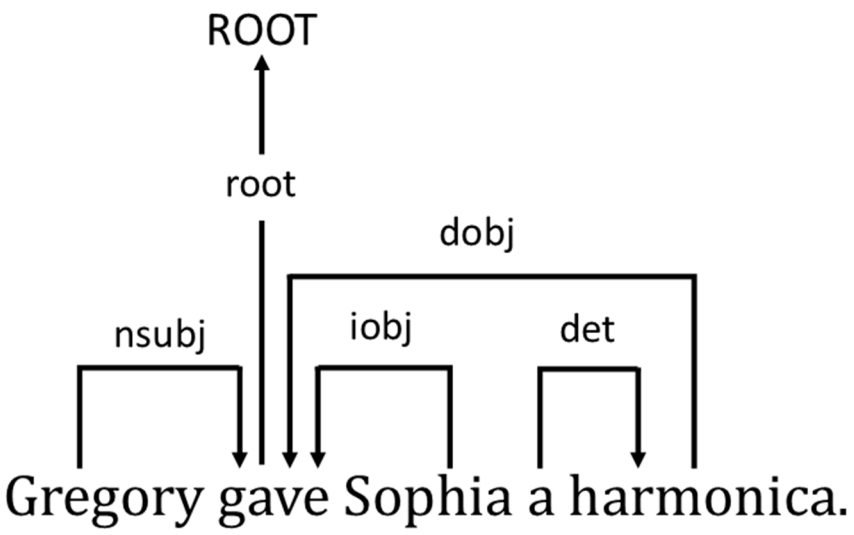

Figure 2. Gave Dependency Tree

The use of syntactic structure to identify meaning, called **syntactic bootstrapping**, has been researched in the fields of psychology and language acquisition. Syntactic bootstrapping was first researched by Brown (1957) and since then it has been demonstrated that toddlers can use syntactic properties such as verb transitivity (Naigles, 1990) and noun count-ness and mass-ness (Soja, 1992) to interpret the meanings of new words. In linguistics, the syntactic constraints of particular verbs are called **c-selection** (short for category selection). So we might think of a verb as having a tree structure sticking out of it. The term **subcategorization frame** is used in distributional semantics and linguistics to refer to the syntactic structure around the verb. In the field of computational semantics, the usefulness of syntactic features for verb clustering and other tasks has been researched and supported by several projects (Sun & Korhonen, 2009; Reichart & Korhonen, 2013). Syntactic features of verbs often come in patterns where a group of verbs will use the same syntactic structure and also have similar

meaning. Because of this, Levin (1993) was able to define very precise verb classes grouped by properties of syntax.

3.3 Lexical Arguments

Another important feature of linguistic context is the lexical arguments taken by each verb. The verb *eat* generally takes a subject that is of some category *person* (or *animate*) and takes a direct object that is of the category *food*. A verb can be thought to define a relationship between a set of arguments. So lexical arguments should theoretically be crucial for inferring verb meaning. In linguistics, the constraints of a verb on the semantic type of arguments they take is called **s-selection** (short for semantic selection), also called **selectional preference**. Both of these terms often refer specifically to noun arguments, which is how they will be used in this paper.

Capturing these features requires a little more work than with syntactic features. When extracting features, we only consider the lemmatized head of the noun phrase. Doing this makes our features much more generalized. Dependency parsing for syntactic features makes this part easy. Keeping only the head noun allows us to throw out irrelevant modifiers and other non-useful information and just keep the most informative word. Keeping the entire noun phrase would make the probability of any feature appearing more than once extremely low. The head noun of the noun phrase allows us to capture the essential semantic type of the argument (except in some cases like "*piece* of cake"). Some verb clustering methods have grouped selectional preferences into semantic categories first, with the use of WordNet or by noun clustering for example, before using them as features to cluster verbs (Sun & Korhonen,

2009; Sun et al., 2013). Part of our goals of future work is to improve on research in selectional preferences, but noun clustering is not a focus of this paper.

4 Modelling Lexical Semantics

In the task of computational lexical semantics, there is a strong motivation to build a more thorough understanding of how word meaning works. Ideally, we would like some way of talking about meaning and semantic relationships that is relevant from a linguistic perspective but is also attainable from a computational perspective. In a computational model of lexical semantics, what is a sense or a category? How can a computer identify hyponymy or polysemy? And how do we identify and understand meaning? Having a way to computationally identify meaning is the ultimate goal of computational lexical semantics. So we have a strong motivation to create and lay out formal ways of talking about these concepts that are relevant both linguistically and computationally.

This section presents a formal abstract model of lexical semantics. This model will be useful for discussing and understanding linguistic topics of lexical semantics from a computational perspective based on some corpus. I will refer to this model as an **instance space**. An instance space is the set of all word instances (of a particular part of speech) from an entire corpus. In other words, every time the verb *eat* is used in a corpus it is a new instance. An instance space is meant to be the space of words from every utterance in a corpus, and so its basic unit is the word instance rather than words which will be considered a broader category. We might even think about the infinite or continuous instance space of all utterances in a language. For convenience and simplicity, we will consider only the instance space of a

particular part of speech. Specifically, I will discuss the **verb instance space** and the **noun instance space** which are two examples.

The instance space model is meant to give us a theoretical approach to how we may think about word meaning and many tasks in lexical semantics from a computational perspective. Meaning is a latent (non-observable) trait of language. So many of the concepts introduced in this section are unsolved problems in Natural Language Processing. But this section will present a broad view of computational lexical semantics and many tasks in it, and it will also present the approach used in this paper for identifying verb clusters as a subtask of this larger schema.

4.1 The Instance Space

An instance space is made up of instances, words, senses, and categories. This section will define these four terms according to the model.

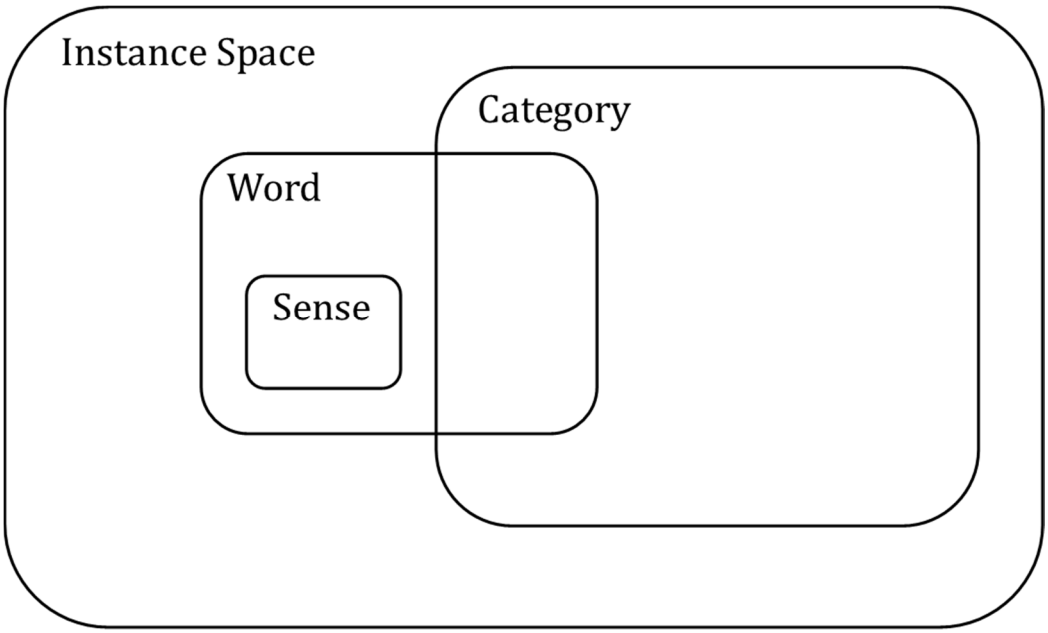

Figure 3. Instance Space

The base unit of an instance space is the instance. So a verb space is made up of verb instances and a noun space is made up of noun instances. An **instance** is a single usage of a word in context. An instance can be defined formally as a lemma L, a part of speech P, a context C, and a meaning M, making it a 4-tuple.

$$I = (L, P, C, M)$$

We might also include other features such morphological form or tense, but for simplicity sake, I will leave these out in this model.

A **lemma** is the uninflected form of a word (*eat* is the lemma of *eating, ate,* or *eaten).* For example:

$$L = \text{“}eat\text{”}$$

A **part of speech** is the syntactic category of an instance. An instance space is categorized by a single part of speech, so all the instance must be of that part of speech. This is a constraint of the model used primarily because different parts of speech (noun, verb) have completely different types of contextual features that can't be easily compared. For example:

$$P = \text{“}verb\text{”}$$

A **context** is all the contextual information of an instance – all of the information associated with an instance that can be used to infer meaning. It may be defined as a set of context features each of which represents one piece of information of the instance's context.

$$C = \{c_1, c_2, c_3, \dots\}$$

A **meaning** is the entire meaning conveyed by an instance in context – all the information intentionally conveyed by the instance in context. It may be defined as a set of

meaning features. Verbs like *swim* and *fly* might share a meaning feature for "motion" but differ in medium of motion "by water" or "by air."

$$M = \{m_1, m_2, m_3, \dots\}$$

Meaning is latent or non-observable which makes identifying it difficult to say the least. Solving this problem completely would theoretically also allow us to solve most tasks in Natural Language Processing. Meaning will be discussed much more in depth in sections (4.2) and (4.3).

Now that we have instance defined, I'll discuss larger structures within the instance space. All of the structures in this section are defined as sets of instances and subsets of the instance space.

A **word**, as I'll define it in this model, is a set of all instances in an instance space that share some lemma.

$$W_l = \{i \in Ispace \mid L = l\}$$

Words in an instance space are disjoint and jointly exhaustive, simply meaning that every instance is contained in exactly one word. Words are observable unlike other structures I will discuss. A consequence of defining a word in this way is that this model does not distinguish homonyms as separate words. Instead these will be treated as different senses of the same word.

A **sense** is a set of all instances in an instance space that have the same lemma and share some aspect of meaning.

$$S_{l,m} = \{\, i \in Ispace \mid L = l, m \subseteq M\}$$

A sense is a subset of some word where all instances share some set of meaning features. Senses do not necessarily have to be disjoint and can overlap. Since senses are defined in terms

of meaning features they are latent (although word sense disambiguation is a field dedicated to finding senses).

A **category** is a set of all instances in an instance space that share some aspect of meaning.

$$C_m = \{ i \in Ispace \mid m \subseteq M \}$$

The words *swim* and *fly* might both be in the category of motion verbs. Given this definition, it seems that senses and categories are extremely similar. A sense is simply the combined restrictions of a word and a category. In fact, a sense is an intersection between a word and a category.

$$S_{l,m} = W_l \cap C_m$$

This tells us that identifying senses and identifying categories are very similar problems. So unsupervised sense disambiguation might be reduced to unsupervised word categorization and vice versa.

4.2 Understanding Meaning

In this section, I'll discuss in detail how we can conceptualize meaning in the instance space model. As meaning is not directly observable, this section is written to outline a theoretical approach to modelling and understanding lexical meaning. The next section presents a more practical approach that is built on theory developed in this section. Meaning as a property of words is a latent (non-observable) trait. In other words, a computer has no way of identifying the meaning of a word or sentence just by reading. Solutions to this problem can

rely on supervised learning based on some annotated model of meaning or can approximate meaning using context and relying on the strong relationship between context and meaning. This project is built on the second approach. This section presents a theoretical structural understanding of meaning and properties derived from meaning to better understand what we are modelling. The next section will discuss how we approximate meaning and its derived properties using context.

In the last section, I defined the meaning of a word as a set of meaning features ($M = \{m_1, m_2, m_3, \dots\}$). This actually makes a few assumptions about the nature of meaning.

1) This model assumes that meaning is compositional – can be broken into smaller meanings.
2) This model assumes that discrete units of meaning exists, units that can't be broken down any further. These are what I've called meaning features.

In terms of the way language is represented in the human brain, either of these hypotheses might be true or might not. Usage based grammar is a linguistic theory famous for hypothesizing that meaning is not separable into autonomous parts and actually exists as one cognitive process (Langacker, 1987). But modelling meaning in this way is very convenient for a theoretical model for two reasons. First, this allows us to relate word meanings in more than one way. Instead of a single number representing the similarity, we could identify that a word is similar to one word by one sub-meaning and to another word by another sub-meaning. So *swim* might be related to *walk* based on motion and *float* based on the domain *water*. This is something that for NLP applications, we would absolutely like to be able to do and something humans seem to be quite good at, so it makes sense to design the model this way. We can

relate this to theories of de-compositionality of meaning that have been proposed by several linguists studying semantics (Jackendoff, 1990; Katz & Fodor, 1963). Second, modelling meaning in this way could allow us to identify what the discrete units of meaning are. Note that this does not necessarily assume a finite set of meaning features in a language. While a corpus is by nature finite and thus will have a finite (but very large) set of meaning features, language in general is an open system and does not necessarily have to be limited in meaning in any way.

4.3 Properties of Words, Senses, and Categories

Next, allow me to discuss the nature of senses and categories and the relationships between them. Note that a sense is simply a special subset of a category and all the conclusions made about senses in this section apply equally well to categories in general. So how do we divide words into senses? Based on the definition above, a sense can be pivoted on any meaning feature or set of meaning features, in other words any aspect of meaning. So unlike words, senses do not have to be disjoint because there can always be a set of instances that meet the conditions of two different senses. Senses can also be hierarchical since within one sense there can always be a more specific sense with the same conditions and additional ones.

There are several properties that are very useful for talking about subsets of an instance space in general (words, senses, or categories). The size or **magnitude** of a subset is the number of instances it contains as a fraction of the total space. If a word makes up 10% of the total instances in a space, then its magnitude is 0.1. This provides a convenient measure of a word's (or sense's or category's) productivity – how much that word is used relative to general language use.

Another characteristic important for this discussion is **specificity**. Specificity is one way of measuring the amount meaning a group of instances contains. Specificity is the number of meaning features held in common by an entire subset. For a sense or category, this will include the meaning condition of that set but might also include other features that are possibly entailed by that meaning. For a word, this is the meaning entailed by the word – held in common by all senses and instances within the word. A word with several completely unrelated senses will have a specificity of 0, which tells us that nothing can be said definitively about the meaning of that word outside of context. Specificity gives a simple, but somewhat unhelpful, measure of how much meaning a group of instances contains.

To define meaning more precisely, we can use a probabilistic distribution. Each subset will be characterized by a **meaning distribution** Ω where each category is a combination of meaning features. The distribution will return the proportionality of instances within the set that have exactly that combination of meaning features.

$$\Omega_M \ = \ P(M|W)$$

This gives us a model for representing the meaning of a word or sense or category including all of their variations in meaning. It also sets up convenient ways of comparing word meanings for similarity which I will discuss below. The meaning distribution of a word (Ω) can be thought of as the word level variant of the meaning of an instance (M). Whereas specificity measures the amount of meaning necessarily entailed by the word, a meaning distribution contains information on all the meanings of all the instances within the word. So while specificity is greater for instances than for words, a meaning distribution contains more information the greater the size of the magnitude.

There are also several linguistic categorizations of meaning and meaning relationships for which we would like formal definitions in this model. We could measure the **synonymy** of two instances as the overlap in meaning between the instances.

$$Synonymy \ = \ |M_1 \cap M_2|$$

We can define the synonymy between two words (or senses or categories) as some metric of distributional similarity between the meaning distributions of words.

$$Synonymy \ = \ sim(\Omega^{(1)}, \Omega^{(2)})$$

The metric of similarity can be flexible. Any measure of similarity between two distributions could work, and there are many different metrics of similarity provided by information theory, statistics, and linear algebra (Jaccard, JS-divergense, cosine similarity, etc.). Note that we might use a similar metric with some asymmetrical measure of similarity (such as KL-divergence) to define hypernymy, where a word is a hypernym of another word if its meaning is a superset of another word's meaning.

One last trait that would be very convenient to measure would be **polysemy**. Light words such as *do, have,* and *make* carry little semantic content by themselves and can be used in a wide variety of contexts to mean a wide variety of things. These words have a high rating of polysemy. So measures of polysemy provide us with a metric for the number of senses a word might have. We can define polysemy in terms of **entropy** over meaning features.

$$H(X) = -\sum_i P(x_i) \log P(x_i)$$

The expression $P(m_i)$ indicates the probability that an instance chosen at random contains the meaning feature m_i. $M$is a list of all possible meaning features in some group of instances. We then get the following equation:

$$Polysemy = -\sum_{i}^{|M|} P(m_i) \log P(m_i)$$

A high entropy indicates that the meaning distribution is spread over a large number of possible meanings, so the word itself entails very little information. In contrast, a low entropy would indicate that a small number of meanings are most likely and so the word itself specifies more information.

While the terms and definitions in this section are primarily theoretical because of the abstract nature of meaning, the next section will explore more concrete ways of approximating these metrics using context.

4.4 Context to Meaning

This section discusses the idea of approximating meaning using context within the instance space model. Section (5.2) will give a more detailed discussion of how we accomplish this with the LDA-based models in this paper. Meaning, senses, and categories are all non-observable properties of language. If there were a computational way of accessing aspects of meaning directly from text, this would solve most of the problems in NLP and much of semantics as a linguistic study. But much of the information of meaning can be approximated by substituting for context. Given the strong relationship between word meaning and word

context, much of the probabilistic structure of word meaning is, theoretically, very similar to the probabilistic structure of context. So we can assign a distribution Λ as follows:

$$\Lambda_C = P(C|W)$$

We can then use Λ as an approximation for Ω and substitute this new distribution in the metrics noted above such as polysemy and synonymy. Similarly, categories and senses could be defined in terms of context features instead of meaning features.

For the work in this paper, we treat each verb as a collection of context features which are modelled probabilistically but not as a single distribution. Instead, we hypothesize latent categories of context features and infer these using LDA as a model. This allows us to treat each verb as a composition of categories of features. These categories can be thought of as analogous to the term category that earlier we defined in terms of meaning. The next section will explain LDA and how we use it to model verbs.

5 Latent Dirichlet Allocation (Beyond Topic Modelling)

In this paper we present two main methods for verb similarity and clustering along with several other previously used methods for comparison. The first method we present in this paper is the use of Latent Dirichlet Allocation (LDA) to identify thematic categories of verb context features, to group verbs into soft clusters (topics), and to define similarities between verbs based on vectors taken from verb proportionality in each topic. The second method presented in this paper is an extension of LDA that we call Recursive LDA which is the use of LDA to categorize verbs just as described above, but done iteratively in a search across multiple

topic splits to develop hierarchical categories of features. This section will outline both of these algorithms and discuss motivations given the characteristics of verbs and context that we are attempting to model. This section will also relate these algorithms to the instance space model discussed in the last section. For this project, we used an implementation of LDA in the MALLET library written by Andrew McCallum and David Mimno.

5.1 A Probabilistic Graphical Model of Documents and Topics

Latent Dirichlet Allocation is a hierarchical graphical model designed for inferring latent thematic categories called topics (Blei et al., 2003). LDA's traditional use is topic modelling but it presents a framework that is useful for identifying latent structure in many domains and applications. LDA has been modified to incorporate syntactic features (Boyd-Graber & Blei, 2008; Griffiths et al., 2004), it has been used to model selectional preferences (Ritter et al., 2010), and it has been modified to work as a language model (Wallach, 2008). The LDA model starts with three major components. LDA models a set of **documents** which is a collection of words or features. **Features** are like word instances in that two features can have the same name but are still considered separate features. Features are observed variables, but each feature is assumed to have some latent topic assignment. **Topics** are hidden thematic categories of related features. Each feature is assigned one topic. A document then has some percentage of each topic based on the number of features with that topic assignment. The function of LDA is to identify these topic assignments, to calculate the proportionality of topics in each document, and to calculate how much a given feature name is associated with a given

topic. In topic modelling, a document is written text, features are words in each document, and topics are themes of the documents. In our model, we will represent each verb as a document containing context features instead of words. This will be described in more detail in section (5.2).

LDA models all of this information using a number of probability distributions with dependencies between, and then it uses some method of Bayesian inference such as Gibbs sampling or variational Bayesian methods to estimate the values of these distributions. This model will be explained thoroughly in this section.

I will discuss LDA's name after I have explained LDA's model when it will make more sense. Here is the diagram of LDA as a graphical model:

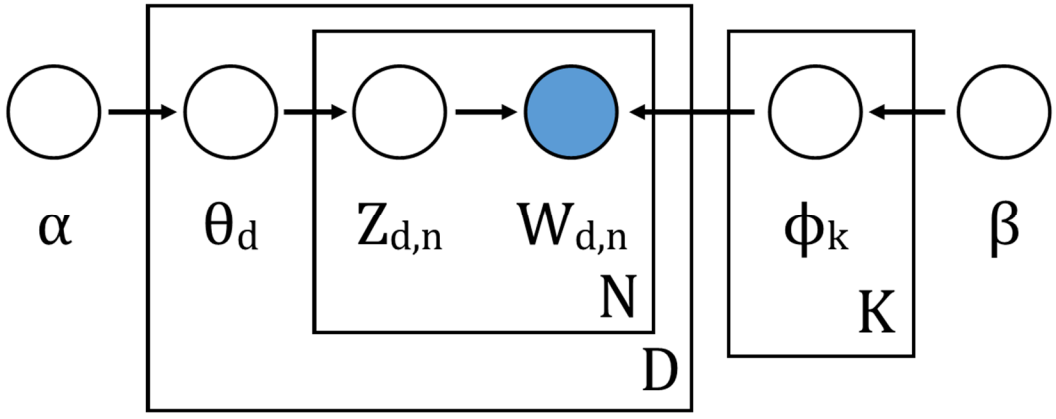

Figure 4. LDA Graphical Model

I will discuss each of these variables. Bear in mind that these terms come from topic modelling but they can be generalized to any of the many applications of LDA.

D – Number of documents

K – Number of topics

V – Number of vocabulary terms

N – Length of a given document in words

Lowercase letters represent indices in these ranges (d is the index for a document in the range 1 … D). A plate in a graphical model designated by some number X indicates that there are X copies of all variables inside that plate. So the variable designated K represents the K topics with K copies of the variable ϕ_k. The D plate represents D documents and the N plate represents a single document with N positions. Arrows represent dependencies.

At the center of the model is the variable $W_{d,n}$ which represents the nth **word** in the dth document. $W_{d,n}$ is the only observable variable in the model. $W_{d,n}$ is an integer value between 1 and V. $W_{d,n}$ represents a word, but it is identified by a number that is an index in the list of vocabulary.

$$W_{d,n} = v \in [1, V]$$

The variable $Z_{d,n}$ is the **topic assignment** of the nth word in the dth document. $Z_{d,n}$ has an integer value between 1 and K, which is an index to the topic of assigned to $W_{d,n}$.

$$Z_{d,n} = k \in [1, K]$$

Each document is composed of some proportionality of each topic. We can model this with a latent probability distribution θ_d. The variable θ_d contains the **document-topic proportionalities**. It is a list of D distributions over K. θ_d for some document d is a distribution of topics. $\theta_{d,k}$ for some document d and some topic k returns the proportion of document d that is topic k. (Some document might be 95% about quantum physics and only 2% percent about baseball). The following equation formalizes the dependency of $Z_{d,n}$ on θ_d showing that $Z_{d,n}$ is sampled from θ_d:

$$\theta_{d,Z_{d,n}} = P(Z_{d,n}|\theta_d)$$

Each topic has some probability of producing each word. ϕ_k contains the **topic-word probabilities**. ϕ_k is a list of K distributions over V. ϕ_k for some topic k is a probability distribution of all vocabulary terms. $\phi_{k,v}$ for some topic k and some word v returns the probability of topic k producing word v. (The topic *baseball* will be much more likely to produce words like *homerun* and *inning* than the topic *quantum physics*). A given topic $Z_{d,n}$ is more likely to produce certain words than others. These likelihoods are specified by ϕ_k. So $W_{d,n}$ is dependent on both $Z_{d,n}$ and ϕ_k. The following equation formalizes this dependency:

$$\phi_{Z_{d,n},W_{d,n}} = P(W_{d,n}|Z_{d,n},\phi_k)$$

Both θ_d and ϕ_k are distributions over a set of mutually exclusive categories. So it makes sense to model these as categorical distributions with $Z_{d,n}$ and $W_{d,n}$ being sampled from them:

$$Z_{d,n} \sim Categorical_K(\theta_d)$$

$$W_{d,n} \sim Categorical_V(\phi_{Z_{d,n}})$$

Of course, these distributions are latent. We have to have some way of inferring them using the data we have. The solution to this is to sample θ_d and ϕ_k each from another distribution. To do this, we will use **Dirichlet** distributions:

$$f(x_1, \ldots, x_k; \alpha_1, \ldots, \alpha_k) = \frac{\Gamma(\sum_{i=1}^{K} \alpha_i)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}$$

The Dirichlet distribution is the conjugate prior of categorical distributions. Given two distributions X and Y, X is a conjugate prior of Y if within the definition of Bayes Rule

$$P(\alpha|\theta) = \frac{P(\theta|\alpha)P(\alpha)}{P(\theta)}$$

X can be used as a prior $p(\alpha)$ that when multiplied by the likelihood function $p(\theta|\alpha)$ produces Y which is a posterior $P(\theta|\alpha)$ in the same family as A. This mathematical property makes it

convenient to use conjugate priors to model other distributions. A Dirichlet distribution can produce the probability of a categorical distribution having a given set of probabilities. A Dirichlet is like a probability distribution of distributions.

We can sample θ_d and ϕ_k from two Dirichlet distributions parameterized by variables α and β (each of which is a vector of values):

$$\theta_d \sim Dirichlet_K(\alpha)$$

$$\phi_k \sim Dirichlet_V(\beta)$$

The variables α and β control the Dirichlet distributions' shapes, affecting the probability of particular sets of probabilities appearing in θ_d and ϕ_k. You can think of the Dirichlet distributions as our hypothesis bias when approaching problems using LDA. We can choose α and β assign certain arrangements of probabilities to be more or less likely, and then we infer other distributions from data and the assigned structure of the model. The advantage of choosing α and β is that we can make distributions sparse. For example, we might like each document to have only one major topic most of the time.

So LDA's name, Latent Dirichlet Allocation, means allocating Dirichlet distributions in order to infer latent distributions.

Now the entire model is set up. The last step is to use some algorithm of Bayesian inference to estimate the distributions we want to find. A common example of a Bayesian inference algorithm used for LDA is Gibbs sampling, also called a Monte Carlo Markov Chain (MCMC). In Gibbs sampling, one variable is estimated at a time in terms of the current values of all the other variables. This is done iteratively in a random process that eventually produces

approximately correct values for variables being estimated. Collapsed Gibbs sampling, a faster version of Gibbs sampling, for LDA gives the following equation:

$$P\left(Z_{d,n} = k \middle| Z^{(-dn)}, W_{d,n} = v, W^{(-dn)}, \alpha\right) \propto (\#Z_k^{d,(-dn)} + \alpha_k) \frac{\#W_v^{k,(-dn)} + \beta_v}{\#W^k + \sum_{v=1}^{V} \beta_v - 1}$$

(where # indicates counts. Ex. $\#W^k$ is the number of words assigned to topic k)

We can use the equation above over a large set of data to estimate $Z_{d,n}$ and use values to estimate the distributions θ_d and ϕ_k.

5.2 LDA with Verbs and Context

Section (5.1) outlined the general model of Latent Dirichlet Allocation. This section will discuss how we used LDA in this paper to model verb context and verb similarities.

We model verbs as documents and context features as words based on the model described above. We treat each verb as a collection of context features. We choose some number of topics to test (these can be tested experimentally). The topics are categories of context features. LDA assigns words that often appear together to the same topics. So using this method, we can group context features together into coherent categories of related features. In other words, features that commonly appear in the same verbs are grouped together more commonly. These categories can be said to correlate with verb classes. Since each verb is composed of multiple topics, they may instead be the components that determine verb classes. Upon observation of the groups formed, this second conclusion seems more likely.

The distribution ϕ_k represents the different topics or categories and their likelihood to produce any given context feature. The distribution θ_d represents all of the verbs and what proportionality of each verb is assigned to each topic. As an example, when LDA is run on

syntactic features split into three major groups – transitive, intransitive, and clause complementizing syntactic features. Each verb then is composed of some proportionality of each group. We can treat each verbs topic proportions as a vector with length equal to the number of topics. We calculate similarities between verbs using Pearson correlation:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

(where $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$)

A number of metrics for similarity can be used in the same way. Pearson performed the best for us experimentally. Once there is a measure of similarity for each pair of verbs, we can produce a verb space. Appendix B contains a few example pictures of this space projected into 3 dimensional space.

5.3 Recursive LDA

As an extension of LDA, we also test a model in which LDA is run iteratively, first on all verbs and then within each topic, to produce a hierarchical structure of verb classes. This could be compared to Hierarchical LDA (hLDA), but is performed in a tree search instead of a distributional process (Blei et al., 2010). A fundamental drawback of LDA is that the number of topics must be chosen in advance. Algorithms that attempt to get around this obstacle are called non-parametric approaches to LDA. Another limitation of LDA is that it assumes a finite set of separate unrelated topics. However, many descriptions of verb classes including Levin's (1993) involve a tree-like structure, where small groups that are related to each other form larger groups.

With this in mind, we hypothesize that the total verb space can be divided into a small number of categories and within each semantic category, verbs relate to each other forming more specific categories. To model this, we run LDA on all verbs with 2 topics. We then divide the verbs into 2 sets where each verb is assigned to the topic with the highest proportionality. Then within each new set of verbs each of which represents a category, we run LDA again using the same features on the smaller set of verbs. This allows us to model smaller categories based on relationships between verbs in a single category. In addition to running LDA with 2 topics, which produces the general tree structure, we also at each partition run LDA with 4 topics. These are used as substitutes of the 2 topic solutions when computing similarity to counter problems of data sparsity. We do this iteratively to produce a tree-like structure of verb-categories. We can model this procedure with the following algorithm.

$LDA(Verbs, Theta, K)$:

($This\ program\ takes\ as\ input$:
$Verbs-\ a\ list\ of\ collections\ of\ features$
$Theta-\ matrix\ of\ verb\ v\ by\ topic\ k, verb-topic\ proportions\ \theta_d\ to\ be\ filled\ in\ by\ LDA$
$K-\ number\ of\ topics$
$The\ program\ uses\ Gibbs\ sampling\ to\ approximate\ \theta_d$
$by\ iteratively\ sampling\ features$
$and\ returns\ values\ in\ the\ variable\ Theta$)

$Recursive\_LDA(Verbs, Theta, iteration)$:

($perform\ Recursive\ LDA\ until\ desired\ number$
$of\ iterations\ x$)
$If\ iteration > x$:
$return$
$end\ if$

$(perform\ LDA\ with\ 4\ topics\ and\ save\ output)$

$LDA(Verbs, Theta, 4)$

$write(Theta)$

$(perform\ LDA\ with\ 2\ topics\ and\ save\ output)$

$LDA(Verbs, Theta, 2)$

$write(Theta)$

$(use\ topic\ split\ to\ divide\ verbs\ and\ run\ Recursive\ LDA\ again\ on\ each$

$new\ set\ of\ verbs)$

$Verbs' = \emptyset$

$Verbs'' = \emptyset$

$for\ v\ in\ |Verbs|:$

    $if\ Theta\ [v][0]\ >=\ 0.5:$

        $Verbs' = Verbs' \cup Verbs[v]$

    $else:$

        $Verbs'' = Verbs'' \cup Verbs[v]$

    $end\ if$

$end\ for$

$Recursive\_LDA(Verbs', Theta, iteration + 1)$

$Recursive\_LDA(Verbs', Theta, iteration + 1)$

To compute a similarity matrix, we can add similarities from each topic from each level in the tree. The results for Recursive LDA given in this paper are based on constituency syntactic features.

6 Methodology

This section will outline our methods of acquiring and processing data. It will discuss the corpus used in this project, how we obtained the list of 3,000 verbs to model, our two methods

of parsing sentences for syntactic features, and what types of features are used in our experiments.

6.1 Corpus

New York Times

The Corpus used in this project is a resource from **New York Times**. It consists of twenty years of New York Times articles from 1987 to 2007. This corpus includes over a million articles including news articles, stock summaries, etc.

Words: 853,753,166

Sentences: 42,833,581

(We have a second larger corpus of Wikipedia articles with 60,000,000 sentences that will be used in future research.)

6.2 List of Verb

The experiments in this paper are based on a list of 3,000 English verbs. The ideal is to have a list of verbs that make up the majority of verb use in English. Parsing our corpus and compiling a list of every word labeled as a verb, produces a list of about 50,000 “verbs”. Actually, most of these are errors and noise – words that were incorrectly labeled by the parser because they did not fit well with any particular part of speech. For example, one early decision

in our process of choosing verbs was to eliminate words that begin with "#" because a large number of interjective hashtags were showing up in our list of verbs.

To refine this list to 3,000 we place a number of constrains on the verbs to use. The first step in this process is that verbs are lemmatized (*ate* and *eaten* are changed to *eat*). We only consider verb strings that start with an alphabetic character followed by one or more alphabetic characters and hyphens ("[A-Za-z][A-Za-z]+"). So verb strings must be at least two characters long, start with a letter, and consist completely of letters and hyphens.

The main constraint that we place on verbs is to include only the top 99% of verb mass. We include the most frequent verbs that make up 99% of verb use. The majority of verbs in the list of 50,000 are infrequent enough that they make up only 1% of verb use. The other 99% is made up of a smaller number of about 3,049 verbs. After this step, a small number of verbs were removed by hand to fix small labeling errors. The most important change is that the verb *be* was removed because this single verb requires a massive amount of memory to process and analyze. The complete list of verbs can be found in Appendix A.

6.3 Parsing Approaches

Given the need for syntactic information as context features, this project relies on two different syntactic parsers. Experiments (1), (2), and (3) in this paper use constituency parsing and experiments (4), (5), and (6) use dependency parsing for syntactic features of context. In this paper, we will compare results from data from each parser to see if either approach is more informative.

Constituency syntactic trees come from Phrase Structure Grammar, a theory of syntax developed by Noam Chomsky (1957). Our constituency parser is provided by Stanford. We use the Factored English Parser, which is a lexicalized syntactic parser (Klein & Manning, 2003).

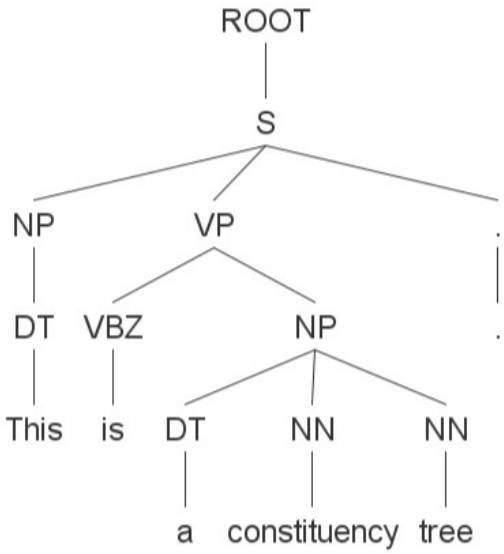

Figure 5. Constituency tree

Phrase structure grammar and constituency parsing have the intention of breaking sentences into hierarchical autonomously functioning phrases called constituents. Each non-terminal in the tree is a phrase node representing the constituent of all the words that are descendants of that node. Constituency trees identify different types of phrases with different identifiers and include rules of how different types of phrases can combine with each other. Constituency trees are most oriented to describe the formation of new sentences and rules thereof. Phrase structure grammar has been able to explain many phenomena of syntax using constituency tree models such as X-bar and others.

Dependency syntactic trees come from dependency grammar theories of syntactic structure. Dependency grammar is less concerned with the formation of phrases and instead is

concerned with syntactic relationships between words. In dependency grammar, there are no phrase nodes. All the nodes in a dependency tree are words or the root node. Each word may have dependents, which are syntactic arguments of that word represented as children. A relation between a word and its dependents is named by the type of syntactic relationship. Our dependency parser is provided by ClearNLP (Choi & McCallum, 2013).

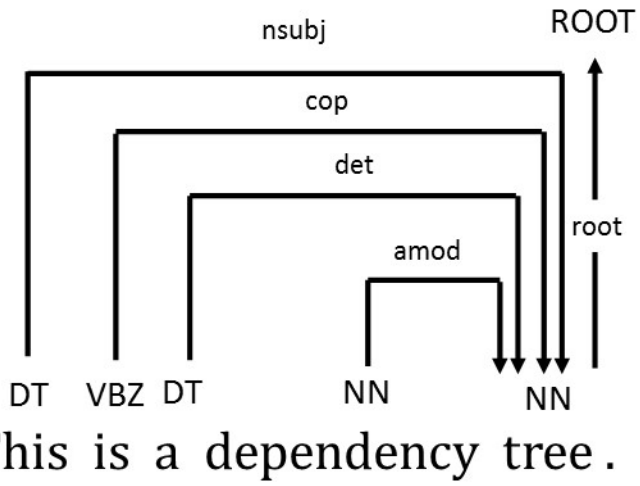

Figure 6. Dependency Tree

Dependency grammar allows for specific naming of relationships between words which opens the possibility for very specifically identified relationships. For many purposes in NLP, this allows for very specific very easily accessible data compared to constituency parsing.

While constituency trees and dependency trees are each more expressive in the characteristic of language syntax that they are designed to convey, there is not much evidence for which structure is a more accurate representation of the mechanisms of natural language syntax in the human brain. The results in this paper comparing constituency and dependency trees might indicate that one structure is more akin to a human meaning acquisition process in

terms of the type of syntactic information used. It might also be indicative of a better or more efficient way of obtaining syntactic features for NLP applications like verb clustering.

6.4 Features

Experiments in this paper compare three different types of features. (1) Purely syntactic features, (2) noun arguments including prepositions and markers for subject, direct object, and indirect object, and (3) A combined set of features of both (1) and (2). Comparing these sets of features can inform us which features are better for verb clustering tasks and also, theoretically, give us insight into which features might be more useful in a human meaning acquisition process. These features also look a bit different depending on whether they are taken from our constituency parser or our dependency parser.

Table 1. Features

Feature	Description	Examples
F1 Constituency Syntax	We consider only 3 nodes under the verb phrase node (VP). A "0" designates that there are fewer than 3 embedded phrase nodes.	0_0_0, NP_0_0, PP_PP_0, NP_PP_PP, NP_ADVP_0
F2 Constituency Lexical	We consider syntactic arguments (or modifiers) of each verb. For any noun argument, we include the string and any syntactic relationship to the verb including specific prepositions. We mark subjects as "S:", direct objects as "DO:", and indirect objects as "IO:".	S:NP(john), S:NP(soap), DO:NP(water), PP(on_NP(behalf)), ADVP
F3 Constituency Syntax + Lexical	This feature set simply conjoins the sets of F1 and F2.	0_0_0, NP_0_0, PP_PP_0, S:NP(john), S:NP(soap), DO:NP(water)

F4 Dependency Syntax	Each feature is the name of a syntactic dependency. Each verb instance can have several dependencies, so each of these counts as a feature.	nsubj, dobj, iobj, prep, agent, acomp, prt, csubj, ccomp
F5 Dependency Syntax + Lexical	For each dependency, if there is a noun argument and or a preposition or particle, we include it in the feature.	prt:up, prep:on:behalf, dobj:water, attr, nsubj:john

7 Experiments & Results

This section lists the experiments in this paper and will introduce our evaluation task. Also, this section will present results for all of the experiments tested for this paper and discuss experiments that should be performed in the future based on these results. Conclusions will be drawn in the next section.

7.1 List of Experiment

Experiment 1 - Syntax from Constituency Parsing

Experiment (1) relies on Pearson correlation between topic associations from LDA (see section 5.2) based on syntactic features from constituency parsing (see F1 in section 6.4).

Example features - *0_0_0, NP_0_0, NP_PP_PP, PP_PP_0*

Experiment 2 - Lexical Arguments from Constituency Parsing

Experiment (2) relies on LDA and Pearson correlation (see section 5.2) based on lexical features from constituency parsing (see F2 in section 6.4).

Example features - *S:NP(john), S:NP(soap), DO:NP(water), PP(on_NP(behalf)), ADVP*

Experiment 3 - Syntax + Lexical Arguments from Constituency Parsing

Experiment (3) relies on LDA and Pearson correlation (see section 5.2) based on syntactic and lexical features from constituency parsing (see F3 in section 6.4).

Example features – *0_0_0, NP_0_0, PP_PP_0, S:NP(john), S:NP(soap), DO:NP(water)*

Experiment 4 - Syntax from Dependency Parsing

Experiment (4) uses syntactic features acquired from dependency parsing (see F4 in section 6.4) modelled using LDA and Pearson correlation (see section 5.2).

Example features - *nsubj, dobj, iobj, prep, agent, acomp, prt, csubj, ccomp*

Experiment 5 - Syntax + Lexical Arguments from Dependency Parsing

Experiment (5) uses syntactic features acquired from dependency parsing (see F5 in section 6.4) modelled using LDA and Pearson correlation (see section 5.2).

Example features - *prt:up, prep:on:behalf, dobj:water, attr, nsubj:john*

Experiment 6 – Recursive LDA with Constituency Syntactic Features

Experiment (6) is our test of Recursive LDA (see section 5.3) and is based on syntactic features extracted from constituency parsing. These results are based off of two layers of Recursive LDA.

Example features - *0_0_0, NP_0_0, NP_PP_PP, PP_PP_0*

Experiment 7 – N-grams

N-grams is a common and well-known statistical language model. In experiment (7), we use features of unigrams, bigrams, and trigrams of words immediately following each verb. N-grams is a fairly simple and easy model of word use, so we use it as a baseline.

Example Features – *3:far:more:water, 2:far:more, 1:far, 3:the:shape:of, 2:the:shape, 1:the*

Experiment 8 – Word2Vec

Word2Vec is on the cutting edge of statistical word models. While it takes the same data-driven approach as n-grams, Word2Vec greatly improves on this approach. It uses skip-grams instead of n-grams to obtain features which makes it possible to obtain more rich data from the same text. It refines its massive sets of data using deep-learning, an unsupervised method of machine learning used for finding salient features. Skip-gram features are similar to n-grams, but in addition to the variable n, skip-grams have a variable indicating the number of words that may be skipped.

Example features – 2-1:far:water, 2-2:the:is, 3-2:this:shape:has

7.2 Triad Evaluation Task

All of our experiments are evaluated against a human intuition task we call a **triad evaluation task**. This task is setup as follows: A single triad is composed of three verbs arranged in a triangle.

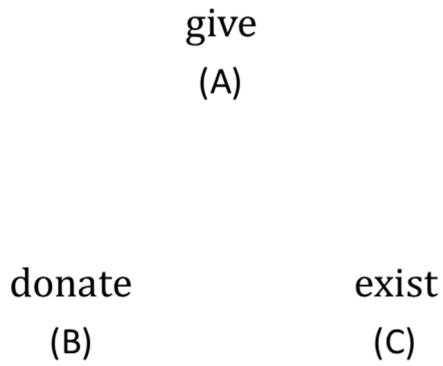

Figure 7.Triad Task

For each triad participants are asked to choose a verb at the bottom of the triad (B for *donate* and C for *exist*) that is most similar in meaning to the verb at the top (*give*). Each participant does this for 96 triads (288 total verbs). The evaluation results in this paper are based on judgments of 29 participants. The same set of triads is used for each participant to allow for an adequate amount of data. Once data has been gathered from this task, the results can be compared to any similarity or distance matrix based on the same verbs. The similarity matrix can be given a percentage of accuracy that is the percentage of triads that agree with the matrix on which verb is closer.

This allows us to test the verb similarities produced by any or our experiments against human judgements, several constraints are placed on verbs chosen for each triad to generally improve judgement reliability of the task. Verbs are chosen to be very close in frequency, to have similar length, and to not start with the same letter as the top verb. This is to assure that participants do not choose verbs by similarity in any of these arbitrary properties instead of meaning.

7.3 Evaluation Results

This section presents the evaluation of each experiment using our triad evaluation task as described above. For experiments (1) to (5), we tested LDA with 3, 4, and 5 topics. Each experiment was used to calculate similarities for 3,000 verbs which are listed in Appendix A. The evaluation for each experiment is based on a sample of 288 verbs with judgements gathered from 29 participants.

Table 2. Evaluation Results

	3 topic	4 topic	5 topic
Constituency Parsing			
Experiment 1. Syntax	0.648936	0.613	0.564
Experiment 2. Lexical Arguments	0.62766	0.670213	0.606383
Experiment 3. Syntax + Lexical	0.553191	0.574468	0.585106
Dependency Parsing			
Experiment 4. Syntax	0.583333	0.604167	0.635417
Experiment 5. Syntax + Lexical	0.59375	0.591398	0.612903
Experiment 6. Recursive LDA	0.614583		
Experiment 7. N-gram	0.57		
Experiment 8. Word2Vec	0.648352		

Inter-rater Reliability	0.739583

Experiment (2), lexical features using constituency parsing, with 4 topics performed with the best evaluation results out of all of our experiments with a rating of 0.67. Experiment (1) with 3 topics tied with Word2Vec for the next best evaluation at 0.65. One interesting observation we could make from this data is that combining syntactic and lexical features seems to hurt evaluation ratings as opposed to keeping them separate.

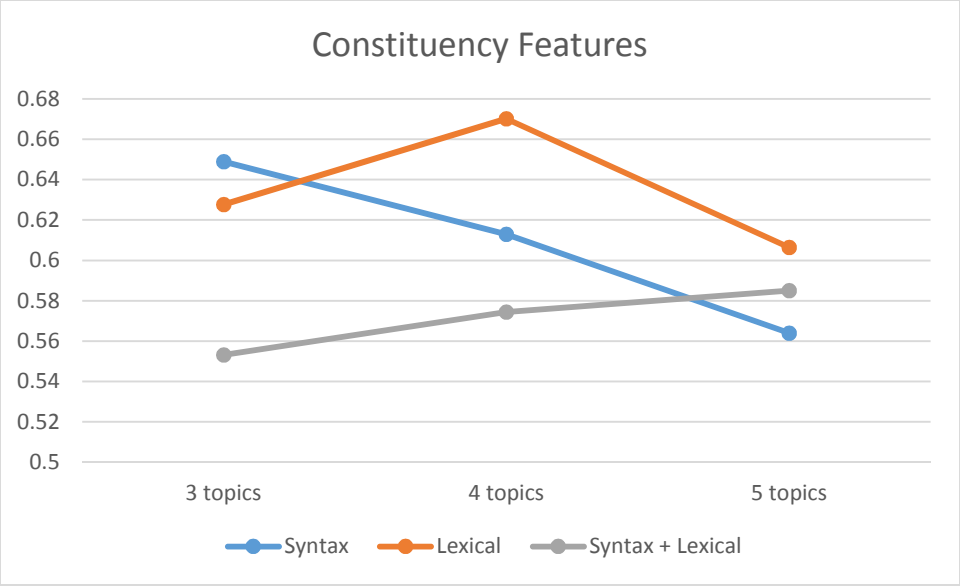

Figure 8. Comparing Constituency Features

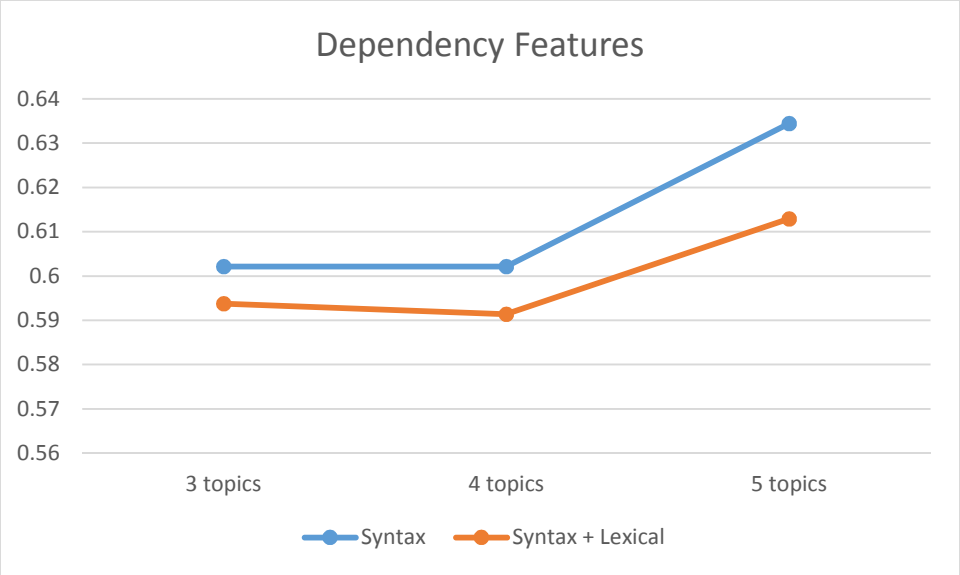

Figure 9. Comparing Dependency Features

In both dependency and constituency based experiments, experiments with either syntactic or lexical features evaluated better than experiments with syntactic and lexical features combined.

The **inter-rater reliability** is our measure of how well one human participant did when their judgement was compared to that of the judgements of the other 28 participants. Inter-rater reliability gives us a measure of how much participants in the evaluation task agreed with each other. It also provides an upper bound on evaluations of verb similarity algorithms. Theoretically, no verb similarity matrix from any experiment should perform better than 0.74, the value of the inter-rater reliability, on our evaluation task.

8 Conclusions

Although the data presented in this section took a significant amount of time and resources to collect, these experiments represent a small set of testable features and parameters. These results show that the right arrangement of linguistic context features

modelled probabilistically with LDA can outperform leading statistical models such as Word2Vec, with still much room for improvement. The best results presented in this paper come from experiment (2) using lexical features from constituency parsing and LDA with 4 topics. This evaluation outperforms Word2Vec which is tied for the second best results. Recursive LDA, while it may present a viable method of producing hierarchical verb classes, evaluated in the mid-range of our experiments. One observation we could is that conjoining syntactic and lexical features is counter-productive according to the results we have. Our experiments which modelled only syntactic features or only lexical features performed better than our experiments with combined features. This suggests that to be able to utilize multiple types of features, we have a strong motivation to develop more viable models for combining these features. Our future research in word modelling and verb clustering will take this insight into account.

9 Future Work

The work presented in this paper is the preliminary findings and models created in the Verbiverse project. In the future, we intend to expand this work to include metrics for the informitivity of a wide variety of features. We also plan to test our models for application in word sense disambiguation tasks. When the Verbiverse project was created, we set out with the goal of expanding on this framework to other parts of speech. The Nouniverse, the Prepiverse, and the Adjectiverse are strong possibilities with the Verbiverse as groundwork. During the development of these projects, we plan on developing metrics of informitivity for

each kind of context information. We plan to develop this both for future NLP applications and to further test which types of context might be part of the human meaning acquisition process.

Bibliography

[1] Blei, D. M., Griffiths, T. L., & Jordan, M. I. (2010). The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. Journal of the ACM, 57(2):1–30.

[2] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. Journal of Machine Learning Research, 3:993–1,022.

[3] Boyd-Graber, J., & Blei, D. M. (2008). Syntactic topic models. In Proceedings of the 22nd Annual Conference on Neural Information Processing Systems (NIPS-08), pages 185–192, Vancouver.

[4] Brown, R. (1957). "Linguistic Determinism and the Part of Speech."

[5] Choi, J., & McCallum, A. (2013). Transition-based Dependency Parsing with Selectional Branching, Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13), 1052-1062, Sofia, Bulgaria.

[6] Chomsky, N. (1957). Syntactic Structures. Mouton de Gruyter, Berlin.

[7] Dang, H. T. (2004). Investigations into the Role of Lexical Semantics in Word Sense Disambiguation. PhD thesis, CIS, University of Pennsylvania.

[8] Dinu, G., & Lapata, M. (2010). Measuring distributional similarity in context. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10), pages 1,162–1,172, Cambridge, MA.

[9] Frith, J. R. (1957). A Synopsis of Linguistic Theory 1930-1955. In Studies in Linguistic Analysis. Oxford Philological Society, Oxford.

[10] Griffiths, T. L., Steyvers, M., Blei, D. M., & Tenenbaum, J. B. (2004). Integrating topics and syntax. In Proceedings of the 18th Annual Conference on Neural Information Processing Systems (NIPS-04), pages 537–544, Vancouver.

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

[12] Jackendoff, R. (1990). *Semantic Structures*. Cambridge, MA: MIT Press.

[13] Joanis, E., Stevenson, S., & James, D. (2008). A general feature space for automatic verb classification. Natural Language Engineering.

[14] Katz, J. J., & Fodor, J. A. (1963). The stucture of a semantic theory. Language, 39(2):170–210.

[15] Kipper, K., Korhonen, A., Ryant, N., & Palmer, M. (2006). Extending VerbNet with Novel Verb Classes. In Proceedings of 5th international conference on Language Resources and Evaluation. Genova, Italy.

[16] Kipper, K., Korhonen, A., Ryant, N., & Palmer, M. (2008). A Large-Scale Classification of English Verbs. In the Journal of Language Resources and Evaluation. 42(1). 21-40.

[17] Kipper-Schuler, K. (2005). VerbNet: A broadcoverage, comprehensive verb lexicon. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, June.

[18] Langacker, R. W. (1987). *Foundations of Cognitive Grammar*, Volume I, Theoretical Prerequisites. Stanford, California: Stanford University Press.

[19] Levin, B. (1993). English verb classes and alternations: A preliminary investigation. Chicago, IL.

[20] Li, B. & Brew, C. (2008). Which are the best features for automatic verb classification. In ACL-08.

[21] Matsuo, Y., Sakaki, T., Uchiyama, K., & Ishizuka, M. (2006). Graph-based word clustering using a Web search engine. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, pages 542–550, Sydney.

[22] Merlo, P., & Stevenson, S., (2001). Automatic Verb Classification Based on Statistical Distributions of Argument Structure. *Computational Linguistics*. 27: 3, 373-408.

[23] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR.

[24] Mikolov, T., Yih, W., & Zweig, G. (2013). Linguistic Regularities in Continuous Space Word Representations. In Proceedings of NAACL HLT.

[25] Miller, G. A. (1995). Wordnet: a lexical database for english. Communications of the ACM, 38(11):39–41.

[26] Naigles, L. (1990). "Children Use Syntax to Learn Verb Meaning". *Journal of Child Language* 17: 357–374.

[27] Reichart, R., & Korhonen, A. (2013). Improved Lexical Acquisition through DPP-based Verb Clustering. In Proceedings of ACL 2013, Sofia, Bulgaria.

[28] Ritter, A., & Etzioni, O. (2010). A latent dirichlet allocation method for selectional preferences. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10), pages 424–434, Uppsala.

[29] Schulte im Walde, S. (2006). Experiments on the automatic induction of german semantic verb classes. Computational Linguistics, 32(2):159–194.

[30] Shi, L., & Mihalcea, R. (2005). Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing. In Proceedings of CICLING.

[31] Soja, N. (1992). "Inferences about the meanings of nouns: the relationship between perception and syntax."

[32] Stevenson, S., & Joanis, E. (2003). Semi-Supervised Verb Class Discovery Using Noisy Features. *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2003)*, 71-78. Edmonton, Alberta, May-June.

[33] Sun, L., & Korhonen, A. (2009). Improving Verb Clustering with Automatically Acquired Selectional Preferences. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). Singapore.

[34] Sun, L., & Korhonen, A. (2011). Hierarchical verb clustering using graph factorization. In EMNLP-11.

[35] Sun, L., Korhonen, A., & Krymolowski, Y. (2008). Verb Class Discovery from Rich Syntactic Data. In Proceedings of the 9th International Conference on Intelligent Text Processing and Computational Linguistics. Haifa, Israel.

[36] Sun, L., McCarthy, D., & Korhonen, A. (2013). Diathesis alternation approximation for verb clustering. In Proceedings of ACL 2013, Sofia, Bulgaria.

[37] Swier, S., & Stevenson, S. (2004). Unsupervised semantic role labelling. In EMNLP-04.

[38] Vlachos, A., Korhonen, A., & Ghahramani, Z. (2009). Unsupervised and constrained Dirichlet process mixture models for verb clustering. In Proceedings of the EACL-09 Workshop on Geometrical Models of Natural Language Semantics (GEMS-09), pages 74–82, Athens.

[39] Wallach, H. M. (2008). Structured topic models for language. Ph.D. thesis, University of Cambridge.

[40] Zapirain, B., Agirre, E., & Marquez, L. (2008). Robustness and generalization of role sets: PropBank vs. VerbNet. In Proc. of ACL.

[41] Zhou, G., Zhao, J., Liu, K., & Cai, L. (2011). Exploiting Web-derived selectional preference to improve statistical dependency parsing. In Proceedings of ACL-11, pages 1,556–1,565, Portland, OR.

Appendix A (List of 3,000 Verbs)

A

abandon, abate, abbreviate, abdicate, abduct, abet, abide, abolish, abort, abound, absolve, absorb, abstain, abuse, abut, accede, accelerate, accent, accentuate, accept, access, acclaim, accommodate, accompany, accomplish, accord, accost, account, accredit, accrue, accumulate, accuse, accustom, achieve, acknowledge, acquaint, acquiesce, acquire, acquit, act, activate, adapt, add, addict, address, adhere, adjoin, adjourn, adjust, administer, admire, admit, admonish, adopt, adore, adorn, advance, advertise, advise, advocate, affect, affiliate, affirm, affix, afflict, afford, age, aggravate, agitate, agree, aid, ail, aim, air, alarm, alert, alienate, align, allay, allege, alleviate, allocate, allot, allow, allude, ally, alter, alternate, amass, amaze, ambush, amend, amount, amplify, amputate, amuse, analyze, anchor, anger, angle, animate, annex, announce, annoy, annul, anoint, answer, antagonize, anticipate, apologize, appal, appeal, appear, appease, applaud, apply, appoint, apportion, appraise, appreciate, apprehend, approach, appropriate, approve, approximate, arch, argue, arise, arm, arouse, arraign, arrange, array, arrest, arrive, articulate, ascend, ascertain, ascribe, ask, aspire, assail, assassinate, assault, assemble, assert, assess, assign, assimilate, assist, associate, assuage, assume, assure, astonish,

attach, attack, attain, attempt, attend, attest, attract, attribute, auction, audit, audition, augment, authorize, autograph, automate, avenge, average, avert, avoid, await, awake, awaken, award

B

back, backfire, baffle, bag, bail, bait, bake, balance, balk, ban, band, bang, banish, bank, bankrupt, baptize, bar, bare, bargain, barge, bark, barricade, base, bash, bask, bat, bathe, batter, battle, bead, beam, bear, beat, beckon, become, bed, befall, befit, befriend, beg, beget, begin, behave, behold, believe, belittle, belong, belt, bemoan, bench, bend, benefit, bequeath, berate, beset, besiege, best, bestow, bet, betray, better, beware, bewilder, bicycle, bid, bill, bind, bite, black, blacken, blame, blanket, blast, blaze, bleach, bleed, blend, bless, blind, blindfold, blink, blister, block, bloom, blossom, blow, bludgeon, blunt, blur, board, boast, bog, boil, bolster, bolt, bomb, bombard, bond, boo, book, boom, boost, boot, border, bore, borrow, botch, bother, bottle, bottom, bounce, bound, bow, bowl, box, boycott, brace, brag, braid, branch, brand, brandish, brave, breach, break, breathe, breed, brew, bribe, bridge, brief, brighten, bring, broadcast, broaden, broker, brood, brown, browse, bruise, brush, bubble, buck, buckle, bud, budget, bug, build, bulge, bulldoze, bully, bump, bundle, buoy, burden, burgeon, burn, burrow, burst, bury, bus, bust, buttress, buy, buzz, bypass

C

calculate, calibrate, call, calm, camouflage, camp, campaign, can, cancel, cap, capitalize, capitulate, captivate, capture, care, carry, cart, carve, cascade, cash, cast, castigate, catalog, catalogue, catapult, catch, categorize, cater, cause, caution, cave, cease, cede, celebrate, cement, censor, censure, center, certify, chain, chair, chalk, challenge, champion, change, channel, chant, characterize, charge, charm, chart, charter, chase, chastise, chat, cheat, check, cheer, cherish, chew, chide, chill, chip, choke, choose, chop, choreograph, christen, chronicle, churn, circle, circulate, circumvent, cite, claim, clamp, clap, clarify, clash, clasp, classify, claw, clean, cleanse, clear, click, climb, clinch, cling, clip, clock, clog, clone, close, clothe, club, cluster, clutch, coach, coalesce, coast, coat, coax, cock, code, codify, coerce, coexist, coil, coin, coincide, collaborate, collapse, collect, collide, colonize, color, comb, combat, combine, come, comfort, command, commandeer, commemorate, commence, commend, comment, commercialize, commission, commit, communicate, commute, compare, compel, compensate, compete, compile, complain, complement, complete, complicate, compliment, comply, compose, compost, compound, comprehend, compress, comprise, compromise, compute, conceal, concede, conceive, concentrate, concern, conclude, concoct, concur, condemn, condense, condition, condone, conduct, confer, confess, confide, configure, confine, confirm, confiscate, conflict, conform, confound, confront, confuse, congratulate, congregate, conjure, connect, conquer, consent, conserve, consider, consign, consist, console, consolidate, conspire, constitute, constrain, constrict, construct, construe, consult, consume, consummate, contact, contain, contaminate, contemplate, contend, contest, continue, contract, contradict, contrast, contribute, contrive, control, convene, converge, converse, convert, convey, convict, convince, cook, cool, cooperate, coordinate, cope, co-produce, copy, copyright, corner, correct, correlate, correspond, corroborate, corrupt, cost, costume, cough, counsel, count, counter, counteract, couple, course, court, cover, covet, crack, craft, cram, cramp, crank, crash, crave, crawl, create, credit, creep, crest, cripple, criticize, critique, crop, cross, crouch, crowd, crown, cruise, crumble, crusade, crush, cry,

crystallize, cull, culminate, cultivate, curb, cure, curl, curry, curse, curtail, curve, cushion, customize, cut, cycle

D

dabble, damage, damn, damp, dampen, dance, dangle, dare, darken, dart, dash, date, dawn, dazzle, deal, debate, debilitate, debunk, decay, deceive, decide, decimate, decipher, deck, declare, declassify, decline, decode, decompose, deconstruct, decorate, decrease, decree, decry, dedicate, deduct, deem, deepen, default, defeat, defect, defend, defer, define, deflate, deflect, defraud, defuse, defy, degenerate, degrade, delay, delegate, delete, deliberate, delight, delineate, deliver, delve, demand, demean, demolish, demonstrate, demoralize, demote, denigrate, denominate, denote, denounce, deny, depart, depend, depict, deplete, deplore, deploy, deport, depose, deposit, depress, deprive, derail, deride, derive, descend, describe, desert, deserve, design, designate, desire, despair, despise, destabilize, destine, destroy, detach, detail, detain, detect, deter, deteriorate, determine, detest, detonate, detract, devalue, devastate, develop, deviate, devise, devolve, devote, devour, diagnose, dial, dictate, die, differ, differentiate, diffuse, dig, digest, digitize, dilute, dim, diminish, dine, dip, direct, disable, disagree, disallow, disappear, disappoint, disapprove, disarm, disavow, disband, disburse, discard, discern, discharge, discipline, disclose, disconnect, discontinue, discount, discourage, discover, discredit, discriminate, discuss, disdain, disenfranchise, disgruntle, disguise, dish, dishearten, disillusion, disintegrate, dislike, dislocate, dislodge, dismantle, dismay, dismiss, disorient, disparage, dispatch, dispel, dispense, disperse, displace, display, dispose, disprove, dispute, disqualify, disregard, disrupt, dissect, disseminate, dissent, dissipate, dissolve, dissuade, distance, distil, distinguish, distort, distract, distress, distribute, distrust, disturb, ditch, dive, diverge, diversify, divert, divest, divide, divorce, divulge, do, dock, document, dodge, dog, dominate, don, donate, doom, dope, dot, double, doubt, douse, down, downgrade, download, downplay, downsize, draft, drag, drain, dramatize, drape, draw, dread, dream, dredge, drench, dress, dribble, drift, drill, drink, drip, drive, droop, drop, drown, drug, drum, dry, dub, duck, duel, dump, dupe, duplicate, dust, dwarf, dwell, dwindle, dye

E

earmark, earn, ease, eat, echo, eclipse, edge, edit, educate, effect, eject, eke, elaborate, elapse, elect, electrify, elevate, elicit, eliminate, elude, email, emanate, embark, embarrass, embed, embellish, embezzle, emblazon, embody, embolden, embrace, embroider, embroil, emerge, emigrate, emit, emphasize, employ, empower, empty, emulate, enable, enact, enamor, encapsulate, encase, enchant, encircle, enclose, encode, encompass, encounter, encourage, encroach, encrust, end, endanger, endear, endorse, endow, endure, energize, enforce, engage, engender, engineer, engrave, engross, engulf, enhance, enjoy, enlarge, enlighten, enlist, enliven, enrage, enrich, enrol, enroll, enshrine, enslave, ensue, ensure, entail, entangle, enter, entertain, enthral, entice, entitle, entrance, entrench, entrust, entwine, envelop, envision, envy, epitomize, equal, equate, equip, eradicate, erase, erect, erode, err, erupt, escalate, escape, eschew, escort, espouse, establish, estimate, etch, evacuate, evade, evaluate, evaporate, even, evict, evidence, evoke, evolve, exacerbate, exact, exaggerate, exalt, examine, exasperate, excavate, exceed, excel, exchange, excise, excite, exclaim, exclude, excuse, execute, exemplify, exempt, exercise, exert, exhaust, exhibit, exhort, exhume, exile, exist, exit, exonerate, expand, expect, expedite, expel, expend, experience, experiment, expire, explain, explode, exploit, explore, export,

expose, expound, express, extend, extinguish, extol, extort, extract, extradite, extrapolate, extricate, exude, eye

F

fabricate, face, facilitate, factor, fade, fail, faint, fake, fall, falsify, falter, fan, fancy, fantasize, fare, farm, fascinate, fashion, fast, fasten, father, fault, favor, fear, feast, feature, feed, feel, feign, fell, fence, fend, ferment, ferry, fertilize, fetch, feud, field, fight, figure, file, fill, film, filter, finalize, finance, find, fine, finish, fire, fish, fit, fix, fixate, flag, flank, flap, flare, flash, flatten, flatter, flaunt, flavor, flee, flesh, flex, flick, fling, flip, flirt, float, flock, flood, floor, flop, flounder, flourish, flow, flower, fluctuate, flush, flutter, fly, focus, foil, fold, follow, foment, fool, foot, forage, forbid, force, forecast, foreclose, foresee, foreshadow, forestall, forfeit, forge, forget, forgive, forgo, form, formalize, formulate, forsake, fortify, forward, foster, foul, found, founder, fracture, frame, freak, free, freeze, frequent, fret, frighten, front, frown, frustrate, fry, fuel, fulfil, fulfill, fumble, function, fund, funnel, furnish, further, fuse

G

gag, gain, gallop, galvanize, gamble, gap, garner, garnish, gather, gauge, gaze, gear, generalize, generate, get, give, glance, glaze, glean, glide, glimpse, glitter, glorify, gloss, glow, glue, gnaw, go, goad, govern, grab, grace, grade, graduate, graft, grant, grapple, grasp, grate, gratify, gravitate, graze, greet, grieve, grill, grind, grip, groom, gross, ground, group, grow, growl, guarantee, guard, guess, guide, gun, gut

H

hack, hail, halt, halve, hammer, hamper, hamstring, hand, handcuff, handicap, handle, hang, happen, harass, harbor, harden, hark, harm, harmonize, harness, harvest, hasten, hatch, hate, haul, haunt, have, hawk, head, headline, heal, heap, hear, heat, heckle, hedge, heed, heighten, help, herald, herd, hesitate, hew, hide, highlight, hijack, hike, hinder, hinge, hint, hire, hit, hitch, hoard, hoist, hold, hole, home, homer, hone, honor, hook, hop, hope, horrify, hospitalize, host, hound, house, hover, howl, hug, hum, humble, humiliate, hunt, hurl, hurry, hurt, hype

I

ice, identify, idle, idolize, ignite, ignore, illuminate, illustrate, imagine, imbue, imitate, immerse, immigrate, immobilize, immortalize, impact, impair, impart, impeach, impede, impel, impersonate, implant, implement, implicate, implode, implore, imply, import, impose, impound, impress, imprint, imprison, improve, improvise, inaugurate, incapacitate, incarcerate, incense, incite, incline, include, incorporate, increase, incriminate, incur, index, indicate, indict, induce, induct, indulge, industrialize, infect, infer, infest, infiltrate, inflame, inflate, inflict, influence, inform, infringe, infuriate, infuse, ingest, inhabit, inhale, inherit, inhibit, initiate, inject, injure, inquire, inscribe, insert, insinuate, insist, inspect, inspire, instal, install, instigate, instil, institute, institutionalize, instruct, insulate, insult, insure, integrate, intend, intensify, interact, intercede, intercept, interest, interfere, interlock, intern, interpret, interrogate, interrupt, intersect, intersperse, intertwine, intervene, interview, interweave, intimidate, intrigue,

introduce, intrude, inundate, invade, invalidate, invent, invert, invest, investigate, invigorate, invite, invoke, involve, iron, irritate, isolate, issue

J

jail, jam, jeopardize, jettison, jog, join, joke, journey, judge, juggle, jump, justify, jut, juxtapose

K

keep, key, kick, kidnap, kill, kiss, kneel, knit, knock, knot, know

L

label, labor, lace, lack, lag, lament, land, landscape, languish, lap, lapse, lash, last, latch, laud, laugh, launch, launder, lay, layer, lead, leak, lean, leap, learn, lease, leave, lecture, legalize, legislate, legitimize, lend, lengthen, lessen, let, letter, level, leverage, levy, liberalize, liberate, license, lick, lie, lift, light, lighten, like, liken, limit, limp, line, linger, link, liquidate, list, listen, litigate, litter, live, load, loan, loathe, lob, lobby, locate, lock, lodge, log, long, look, loom, loop, loosen, loot, lose, love, lower, lumber, lump, lure, lurk

M

magnify, mail, maim, maintain, major, make, malfunction, man, manage, mandate, maneuver, manifest, manipulate, manufacture, map, mar, march, marginalize, mark, market, marry, marshal, marvel, mash, mask, masquerade, mass, massacre, master, mastermind, match, mate, materialize, matter, mature, maximize, mean, meander, measure, meddle, mediate, meditate, meet, meld, melt, memorialize, memorize, menace, mend, mention, mentor, merge, merit, mesh, mesmerize, mess, mete, migrate, milk, mill, mimic, mind, mine, mingle, minimize, minister, mint, mire, mirror, mishandle, misidentify, misinterpret, mislead, misplace, misrepresent, miss, misspell, mistake, mistreat, misunderstand, misuse, mitigate, mix, mobilize, mock, model, moderate, modernize, modify, modulate, mold, molest, monitor, monopolize, moor, mop, morph, motivate, motorize, mount, mourn, move, mow, multiply, murder, muse, muster, mutate, mute, mutilate

N

nail, name, narrate, narrow, nationalize, navigate, near, necessitate, need, negate, neglect, negotiate, neighbor, nest, nestle, net, neutralize, nickname, nod, nominate, notch, note, notice, notify, nourish, nullify, number, nurse, nurture

O

obey, object, obligate, oblige, obliterate, obscure, observe, obsess, obstruct, obtain, occupy, occur, offend, offer, officiate, offset, oil, omit, ooze, open, operate, oppose, oppress, opt, orbit, orchestrate, ordain, order, organize, orient, originate, orphan, ostracize, oust, outdo, outfit, outgrow, outlast, outlaw,

outline, outlive, outnumber, outpace, outperform, outrage, outrun, outscore, outsource, outstrip, outweigh, overcome, overestimate, overflow, overhaul, overhear, overheat, overlap, overload, overlook, overpower, override, overrule, overrun, oversee, overshadow, overstate, overtake, overthrow, overturn, overwhelm, owe, own

P

pace, pack, package, paddle, page, paint, pair, pan, panel, panic, parade, parallel, paralyze, paraphrase, pardon, pare, park, parody, parole, parse, part, participate, partner, party, pass, paste, patch, patent, patrol, patronize, pattern, pause, pave, pay, peak, peddle, peel, peer, peg, pelt, pen, penalize, penetrate, pepper, perceive, perch, perfect, perform, perish, permeate, permit, perpetrate, perpetuate, persecute, persevere, persist, personify, persuade, pertain, pervade, petition, phase, phone, photograph, phrase, pick, picket, pickle, picture, piece, pierce, pile, pilot, pin, pinch, pinpoint, pioneer, pipe, pit, pitch, placate, place, plague, plan, plant, plaster, plate, play, plead, please, pledge, plot, plow, pluck, plug, plummet, plunder, plunge, ply, poach, pocket, point, poise, poison, poke, polarize, police, polish, politicize, poll, pollute, ponder, pool, pop, popularize, populate, portray, pose, posit, position, possess, post, postpone, pound, pour, power, practice, praise, pray, preach, precede, precipitate, preclude, predate, predecease, predicate, predict, predominate, prefer, preoccupy, prepare, presage, prescribe, present, preserve, preside, press, pressure, presume, pretend, prevail, prevent, preview, prey, price, pride, prime, print, privatize, prize, probe, proceed, process, proclaim, procure, prod, produce, profess, profile, profit, program, progress, prohibit, project, proliferate, prolong, promise, promote, prompt, promulgate, pronounce, prop, propagate, propel, propose, prosecute, prosper, protect, protest, protrude, prove, provide, provoke, prune, pry, publicize, publish, pull, pulsate, pulse, pump, punch, punctuate, puncture, punish, punt, purchase, purge, purify, purport, pursue, push, put, puzzle

Q

quadruple, qualify, quantify, quarrel, quarterback, quash, quell, question, quicken, quiet, quilt, quip, quit, quote

R

race, rack, radiate, radio, rage, raid, rail, rain, raise, rake, rally, ram, ramble, range, rank, ransack, rap, rape, rat, rate, ratify, ration, rationalize, rattle, ravage, rave, raze, reach, react, read, ready, reaffirm, realign, realize, reap, reappear, rear, rearrange, reason, reassemble, reassert, reassess, reassign, reassure, rebel, rebound, rebuff, rebuild, rebuke, rebut, recall, recant, recapture, recast, recede, receive, recess, recharge, recite, reckon, reclaim, recline, recognize, recoil, recommend, reconcile, reconfigure, reconsider, reconstitute, reconstruct, reconvene, record, recount, recoup, recover, recreate, re-create, recruit, rectify, recuperate, recur, recycle, red, redeem, redefine, redesign, redevelop, redirect, rediscover, redistribute, redo, redress, reduce, reel, re-emerge, re-enter, re-establish, re-evaluate, re-examine, refer, refill, refine, reflect, refocus, reform, refrain, refresh, refrigerate, refund, refurbish, refuse, refute, regain,

regard, register, regret, regroup, regulate, rehabilitate, rehearse, rehire, reign, reignite, reimburse, rein, reinforce, reinstall, reinstate, reinterpret, reintroduce, reinvent, reinvigorate, reissue, reiterate, reject, rejoice, rejoin, rejuvenate, rekindle, relate, relax, relay, release, relegate, relent, relieve, relinquish, relish, relive, relocate, rely, remain, remake, remark, remarry, remedy, remember, remind, reminisce, remodel, remove, rename, rend, render, renege, renegotiate, renew, renounce, renovate, rent, reopen, reorganize, repackage, repaint, repair, repatriate, repay, repeal, repeat, repel, replace, replay, replenish, replicate, reply, report, reposition, represent, repress, reprimand, reprint, reprise, reproduce, repudiate, repurchase, repute, request, require, reroute, reschedule, rescind, rescue, research, resell, resemble, resent, reserve, reset, resettle, reshape, reside, resign, resist, resolve, resonate, resort, respect, respond, rest, restart, restate, restore, restrain, restrict, restructure, result, resume, resurface, resurrect, resuscitate, retail, retain, retake, retaliate, retell, rethink, retire, retool, retort, retrace, retract, retrain, retreat, retrieve, return, reunite, reuse, revamp, reveal, revel, revere, reverse, revert, review, revile, revise, revisit, revitalize, revive, revoke, revolt, revolutionize, revolve, reward, rework, rewrite, rezone, rhyme, rid, ride, ridicule, rig, right, ring, rinse, rip, ripen, rise, risk, rival, rivet, roam, roar, roast, rob, rock, roll, room, root, rope, rot, rotate, round, rouse, rove, row, rub, ruin, rule, rumble, rumor, run, rupture, rush, rust

S

sabotage, sack, sacrifice, sadden, saddle, safeguard, sag, sail, salt, salute, salvage, sample, sanction, sandwich, sap, satirize, satisfy, saturate, save, saw, say, scale, scan, scar, scare, scatter, schedule, school, scoff, scold, scoop, scorch, score, scorn, scour, scout, scramble, scrap, scrape, scratch, scream, screen, screw, script, scroll, scrub, scrutinize, sculpt, sculpture, scuttle, seal, search, season, seat, secede, secure, seduce, see, seed, seek, seem, seep, segregate, segue, seize, select, sell, send, sense, sentence, separate, sequence, sequester, serve, service, set, settle, sever, sew, shade, shadow, shake, shame, shape, share, sharpen, shatter, shave, shear, sheathe, shed, shell, shelter, shelve, shepherd, shield, shift, shimmer, shine, ship, shock, shoot, shop, shore, short, shorten, shoulder, shout, shove, show, showcase, shower, shred, shrink, shroud, shrug, shuffle, shun, shut, shutter, shuttle, shy, sicken, side, sideline, sift, sign, signal, signify, silence, simmer, simplify, simulate, sing, single, sink, siphon, sit, situate, size, skate, sketch, skew, ski, skim, skip, skirt, skyrocket, slam, slant, slap, slash, slate, slaughter, slay, sleep, slice, slide, slip, slit, slope, slow, slug, slump, smack, smash, smear, smell, smile, smite, smoke, smooth, smother, smuggle, snag, snap, snatch, sneak, sniff, snow, snub, soak, soar, sob, socialize, soften, solicit, solidify, solve, soothe, sort, sound, sour, sow, space, span, spar, spare, spark, spawn, speak, spearhead, specialize, specify, speculate, speed, spell, spend, spew, spice, spike, spill, spin, spiral, spit, splash, splice, splinter, split, spoil, sponsor, sport, spot, spotlight, sprain, sprawl, spray, spread, spring, sprinkle, sprint, sprout, spur, spurn, spy, squander, square, squash, squat, squeeze, stab, stabilize, stack, staff, stage, stagger, stagnate, stain, stake, stalk, stall, stamp, stand, standardize, star, stare, start, startle, starve, state, station, stave, stay, steal, steam, steep, steer, stem, step, stereotype, sterilize, stick, stiffen, stifle, stigmatize, stimulate, sting, stipulate, stir, stitch, stock, stockpile, stoke, stomp, stone, stoop, stop, store, storm, straddle, straighten, strain, strand, strangle, strap, stray, streak, stream, streamline, strengthen, stress, stretch, strew, strike, string, strip, stripe, strive, stroke, stroll, structure, struggle, strut, stud, study, stuff, stumble, stump, stun, stunt, style, stylize, stymie, subdivide, subdue, subject, submerge,

submit, subpoena, subscribe, subside, subsidize, substantiate, substitute, subtitle, subtract, subvert, succeed, succumb, suck, sue, suffer, suffice, suffocate, suffuse, suggest, suit, sum, summarize, summon, superimpose, supersede, supervise, supplant, supplement, supply, support, suppose, suppress, surf, surface, surge, surmise, surmount, surpass, surprise, surrender, surround, survey, survive, suspect, suspend, sustain, swallow, swamp, swap, swarm, sway, swear, sweat, sweep, sweeten, swell, swerve, swim, swing, swipe, swirl, switch, swivel, swoop, symbolize, sympathize, synchronize, syndicate, synthesize

T

tabulate, tack, tackle, tag, tail, tailor, taint, take, talk, tally, tame, tamper, tan, tangle, tap, tape, taper, target, tarnish, taste, tattoo, taunt, tax, teach, team, tear, tease, teem, telephone, televise, tell, temper, tempt, tend, tender, term, terminate, terrify, terrorize, test, testify, tether, texture, thank, thaw, theorize, thicken, thin, think, thrash, thread, threaten, thrill, thrive, throw, thrust, thwart, tick, ticket, tie, tighten, tile, tilt, time, tinge, tint, tip, tire, title, toast, toil, tolerate, tone, top, topple, torment, torture, toss, total, touch, tour, tout, tow, tower, toy, trace, track, trade, traffic, trail, train, trample, transcend, transcribe, transfer, transform, translate, transmit, transpire, transplant, transport, trap, trash, traumatize, travel, traverse, tread, treasure, treat, trek, tremble, trespass, trick, trickle, trigger, trim, trip, triple, triumph, trouble, trounce, trump, trust, try, tuck, tumble, tune, turn, tutor, tweak, twist, type, typify

U

uncover, undercut, underestimate, undergo, underlie, underline, undermine, underperform, underscore, understand, understate, undertake, underwrite, undo, undulate, unearth, unfold, unify, unite, unleash, unload, unlock, unravel, unseat, unveil, up, update, upgrade, uphold, uplift, uproot, upset, urge, urinate, use, usher, usurp, utilize, utter

V

vacate, vacation, vaccinate, validate, value, vandalize, vanish, vanquish, vary, vault, veer, veil, vend, vent, venture, verify, verse, vest, vet, veto, vibrate, victimize, videotape, vie, view, vilify, vindicate, violate, visit, visualize, voice, void, volunteer, vomit, vote, vow

W

wade, wage, wager, wail, wait, waive, wake, walk, wall, wander, wane, want, war, ward, warm, warn, warp, warrant, wash, waste, watch, water, wave, waver, wax, weaken, wean, wear, weather, weave, wed, wedge, weed, weep, weigh, weight, welcome, weld, whale, wheel, whip, whisk, whisper, whistle, widen, widow, wield, will, win, wind, wipe, wire, wish, withdraw, wither, withhold, withstand, witness, wonder, woo, word, work, worry, worsen, worship, wound, wrap, wreak, wreck, wrest, wrestle, wrinkle, write

Y

yearn, yell, yield

Z

zone, zoom

Appendix B (Verb Space Images)

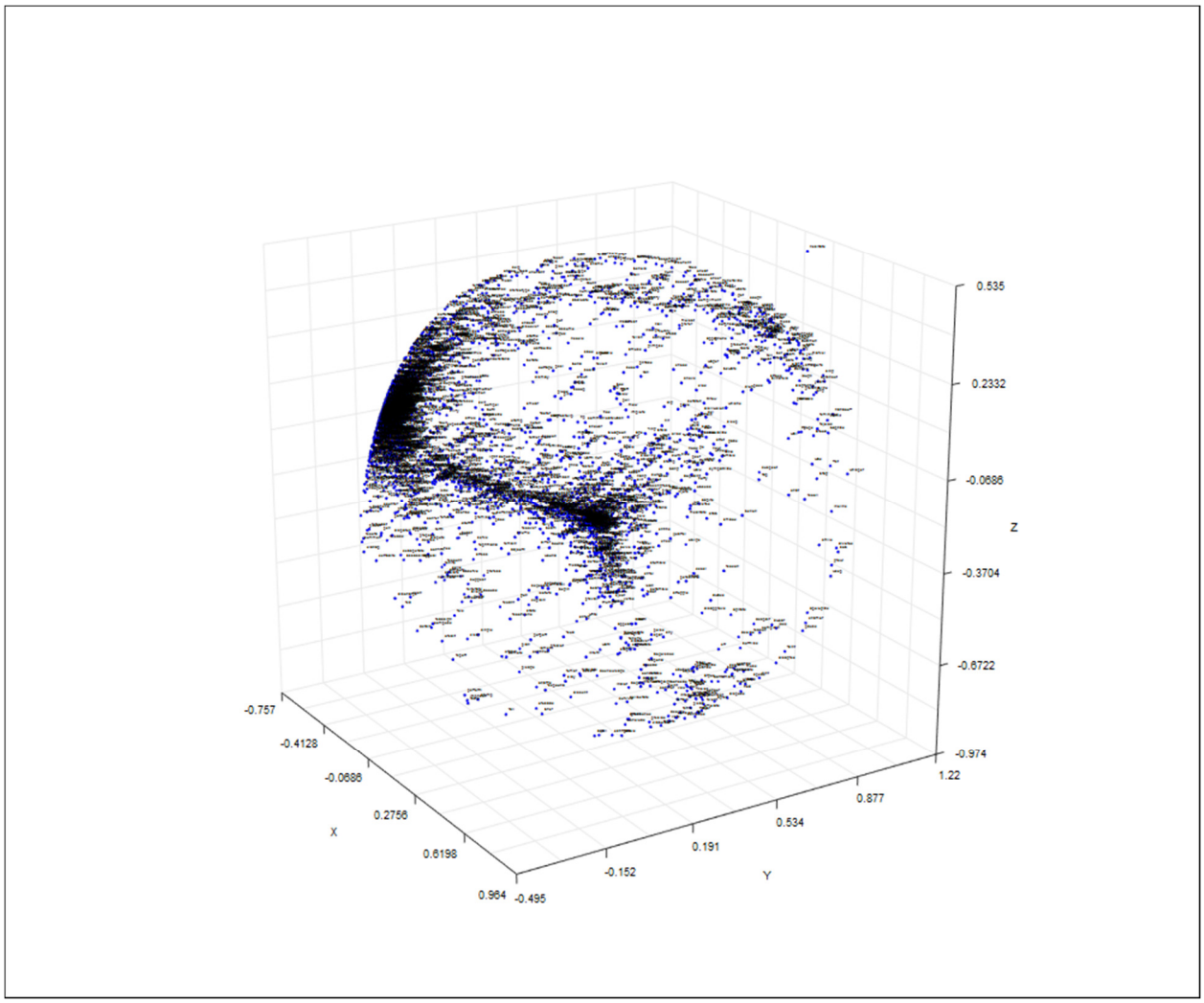

Figure 10. Verb Space

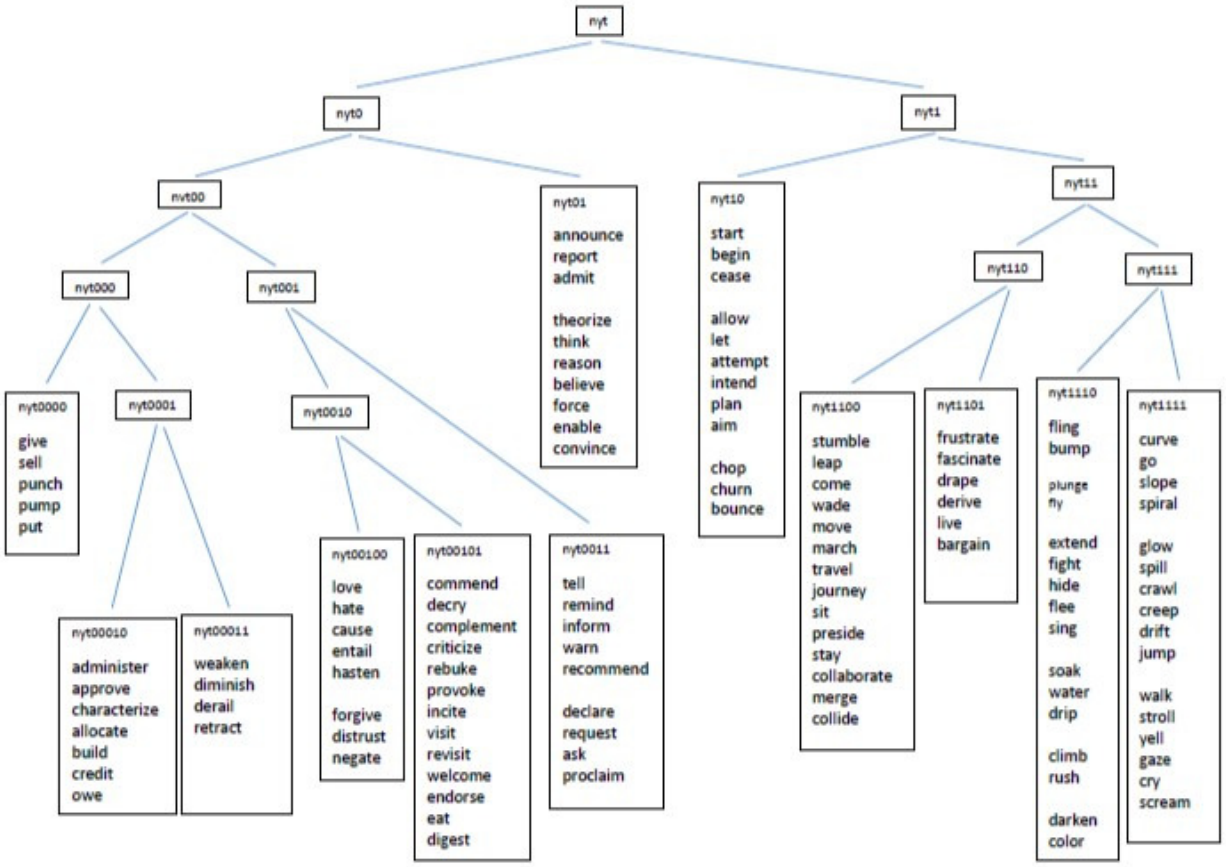

Figure 11. Verb Classes from Recursive LDA

Figure 12. Think/Believe Verbs

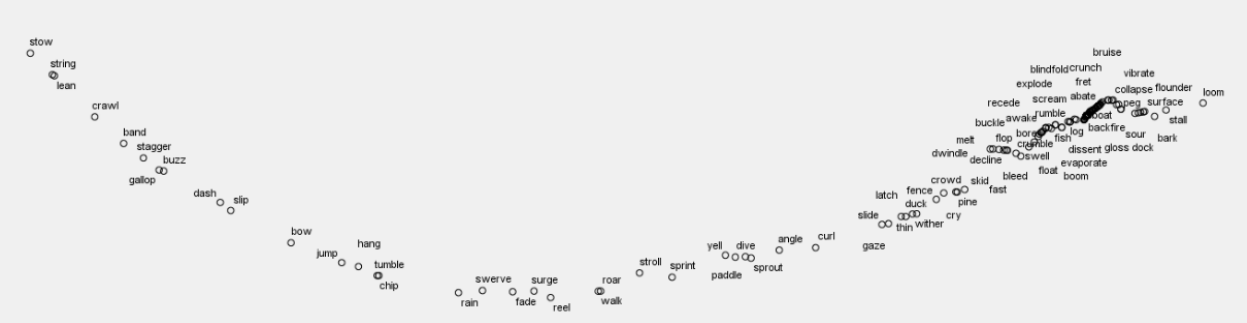

Figure 13. Manner-of-Motion Verbs

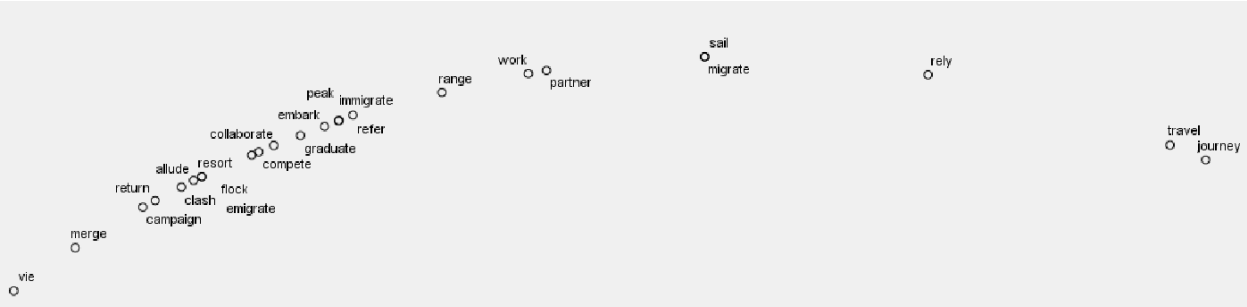

Figure 14. Other Motion Verbs

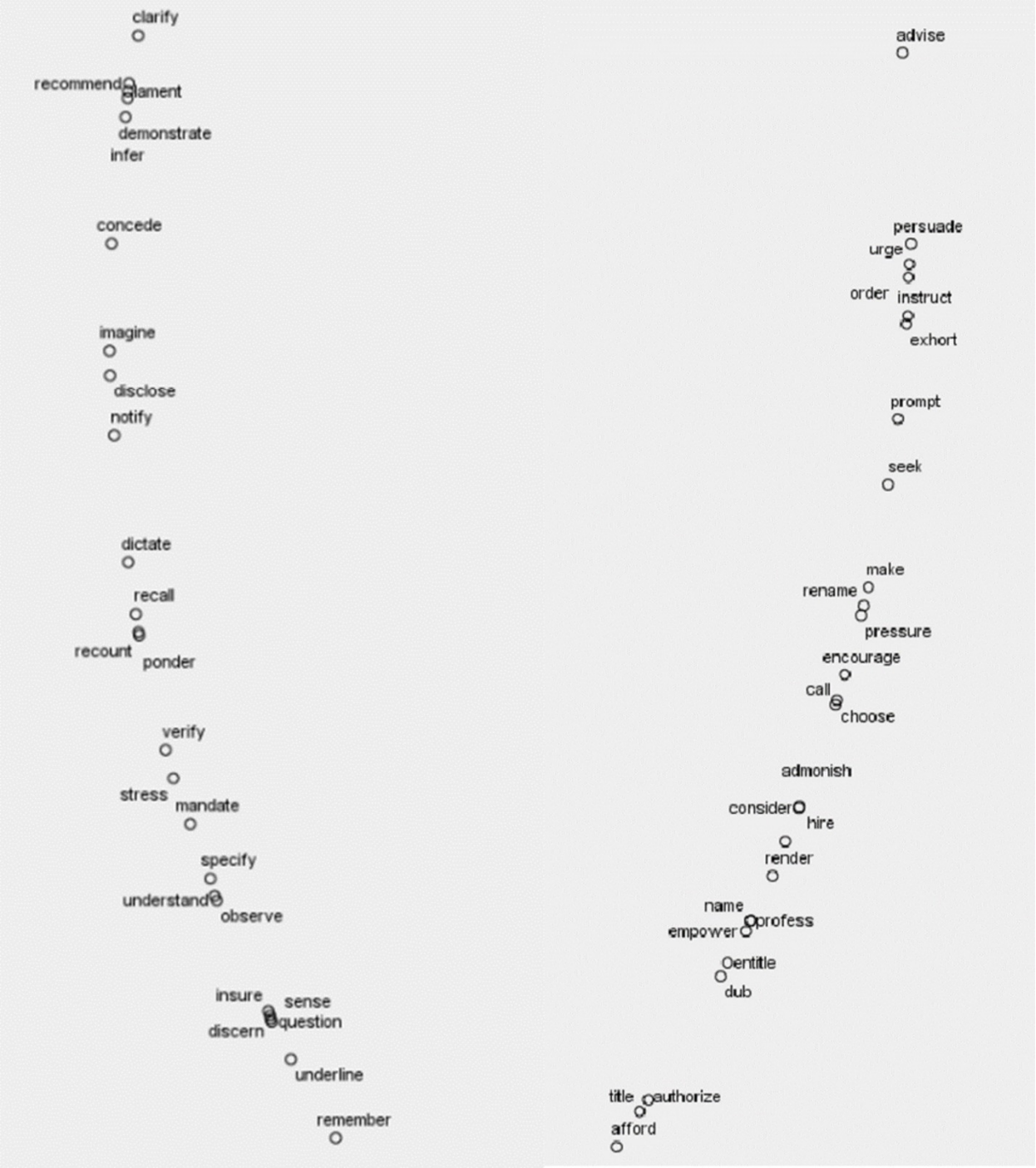

Figure 15. Mental-State Verbs