PROBABILISTIC MODELING OF VERBNET CLUSTERS

by

DANIEL WYDE PETERSON

B.A. Mathematics, University of Wyoming, 2009

M.S. Electrical Engineering, University of Wyoming, 2010

M.S. Computer Science, University of Colorado at Boulder, 2014

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirement for the degree of

Doctor of Philosophy

Department of Computer Science

2019

This thesis entitled:

Probabilistic Modeling of VerbNet Clusters

written by Daniel Wyde Peterson

has been approved for the Department of Computer Science

Martha Palmer

James Martin

Date: _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Abstract

Peterson, Daniel Wyde (Ph.D., Computer Science)

Probabilistic Modeling of VerbNet Clusters

Thesis directed by Professor Martha Palmer

The objective of this research is to build automated models that emulate VerbNet, a semantic resource for English verbs. VerbNet has been built and expanded by linguists, forming a hierarchical clustering of verbs with common semantic and syntactic expressions, and is useful in semantic tasks. A major drawback is the difficulty of extending a manually-curated resource, which leads to gaps in coverage. After over a decade of development, VerbNet has missing verbs, missing senses of common verbs, and is missing appropriate classes to contain at least some of them. Although there have been efforts to build VerbNet resources in other languages, none have received as much attention, so these coverage issues are often more glaring in resource-poor languages. Probabilistic models can emulate VerbNet by learning distributions from large corpora, addressing coverage by providing both a complete clustering of the observed data, and a model to assign unseen sentences to clusters. The output of these models can aid the creation and expansion of VerbNet in English and other languages, especially if they align strongly with known VerbNet classes.

This work develops several improvements to the state-of-the-art system for verb sense induction and VerbNet-like clustering. The baseline is two-step process for automatically inducing verb senses and producing a polysemy-aware clustering, that matched VerbNet more closely than any previous methods. First, we will see that a single-step process can produce

better automatic senses and clusters. Second, we explore an alternative probabilistic model, which is successful on the verb clustering task. This model does not perform well on sense induction, so we analyze the limitations on its applicability. Third, we explore methods of supervising these probabilistic models with limited labeled data, which dramatically improves the recovery of correct clusters. Together these improvements suggest a line of research for practitioners to take advantage of probabilistic models in VerbNet annotation efforts.

Acknowledgements

This work has been supported by my employment as a data scientist, and I would like to formally thank the companies that helped pay tuition and supported my commitment to finishing this research.

TravelShark (2013-2014) was the first to offer me full-time employment with support for my PhD, and I learned a lot about solving problems from my immediate supervisor and mentor, Ambarish Jash. Owen Robertson and Toma Bedolla also deserve my gratitude.

TrustYou (2014-2017) offered me a position and allowed me to work remotely from home, never requiring I move to be near the company headquarters in Germany. I am especially grateful to CTO Steffen Wenz, but I received moral support from the other data scientists and managers. Thanks are due to Jakob Riegger, Benjamin Jost, Miguel Cabrera, and Nicholas Gow.

Oracle (2017-present), and the Machine Learning Research Group in Oracle Labs, have been instrumental in my actual completion of this program. I feel priveleged to work in an environment with such fantastic researchers, and humbled by the amount of support they have given me through this process, including workshopping my papers and presentations to help me get my message through clearly. Pallika Kanani, Adam Pocock, Tori Orr, Jean-Baptiste Tristan, Michael Wick, and Rob Oberbreckling deserve particular mention, and without the approval from managers Stephen Green and Karl Haberl to pursue and publish experiments in supervised topic models, it is unlikely I would have been able to complete this work.

On a personal note, I need to thank my wife, Angela. I feel beloved, every day, and there is no way I can express the impact that has on my life. I must thank my parents, Dave and Kathleen, who raised me and helped me grow, and who have been proud of me and my accomplishments. And I need to thank my sisters and friends and cousins and neighbors, all of whom have given emotional support that helped me carry on.

Contents

List of Figures

List of Tables

Chapter 1

Contributions

Natural language processing (NLP), or natural language understanding, is a field broadly concerned with enabling computers to interact with human (or "natural") language. One of the most difficult open problems in natural language processing is how to model meaning in a way that permits computers to generalize and reason about the text. This sub-field is called computational semantics.

VerbNet (Kipper-Schuler, 2005) is a resource that is based around semantic (meaning-based) groups of verbs. It aids computer understanding of natural language, because verbs in the same group tend to share important aspects of meaning, and also tend to be expressed in a small number of possible sentence structures (called syntactic frames; sets of meaning-preserving syntactic frames may be called syntactic alternations). VerbNet provides useful semantic information to aid in natural language processing tasks, but it suffers from a chronic problem. It does not have enough coverage. Manually adding new verbs and classes to VerbNet is a challenging task requiring linguistic expertise, and is likely infeasible to carry out for every potential domain of application, or for multiple languages. Addressing this coverage issue is the major motivator of this research, and VerbNet is explained in more detail in Section 2.2.3

The following pages describe an effort to improve VerbNet by expanding coverage, using

large bodies of text to create the verb clusters. A baseline framework from Kawahara et al. (2014b), which is explained in detail in Chapter 3, builds VerbNet-like clusters from observed dependency parses in a large corpus. This is a step-wise framework, that breaks the verb clustering process down into steps of sense induction (Section 2.4.1) and verb clustering (Section 2.4.2). The clusters from this baseline have higher VerbNet alignment than any previous method. Systematically experimenting on this baseline model has produced a set of independent improvements to the state of the art. Each aids in understanding of the task, and brings us closer to achieving an automatic, data-driven extension to VerbNet, to bridge the gap in coverage for any corpus. These improvements are explained in detail in Chapters 3, 4 and 5.

1.1 Hypotheses

The foundational hypothesis that drives this work was laid down by Levin (1993). We state it as:

Hypothesis 1 *Observable syntactic behavior is a reflection of a verb's semantics.*

Levin introduced a set of semantic classes for verbs, which gave evidence of this hypothesis in action. Each class was semantically coherent, in the sense that the verbs all shared important semantic aspects. Further, each class had a set of syntactic patterns or allowable alternations, that were distinct. To illustrate this principle, consider the example verbs "break" and "cut." The sentences "John broke the window," and "John cut the bread," are syntactically identical and both are grammatically acceptable. But, while "The window broke," is a fine sentence that may be observed in a corpus, "The bread cut," is an awkward construction and is unlikely to occur. This is evidence that "break" and "cut" belong in different semantic classes. However, "shatter" can participate in both syntactic patterns, and is much more semantically similar to "break".

Levin's classes gave significant evidence of Hypothesis 1, but did not provide enough coverage to be used for computational semantics. VerbNet, which is a major focus of this work, expanded these original classes, adding thousands more verbs and many new classes, as well as rich semantic annotation.

We adopt Hypothesis 1 because it is plausible, and can be experimentally validated. Levin's classes and VerbNet have stood up to scrutiny in the field of linguistics and have been useful to NLP practitioners. The even more important aspect is that it is testable. The organizing principle is that verbs with the same semantics share syntactic patterns. Syntactic patterns are observable in a large corpus (as detailed in Chapter 2.1). We can test the hypothesis by grouping together verbs that share syntax, and test whether the resulting clusters are semantically coherent. And, we can test whether verbs that share semantics do, in fact, share clusters.

In this work, we often use VerbNet or related semantic verb classes to evaluate our success. VerbNet is a large-scale semantic grouping of verbs, based on Levin's verb classes, so it provides many examples of semantic verb clusters with similar syntactic preferences. We aim for our models to produce those same clusters, as correctly as possible.

Hypothesis 2 *Sense disambiguation is a crucial component of deriving semantic groups of verbs.*

Polysemy is a relationship where the same word can have multiple, partially related meanings. For example, "enter a room" has a different meaning than "enter university," and "cutting prices" is different from "cutting ribbon." Homonymy is a relationship where words have the same spelling but completely unrelated meanings, such as "bolting in terror" or "bolting steel". This work considers all meanings for the same word as distinct senses, and in general we will not distinguish between polysemy and homonymy. Some highly-frequent verbs have a large number of senses, and to put those senses together into the same class is often entirely incorrect. "Bolt" sometimes shares semantics with "attach", "staple", or "screw", and sometimes "bolt" means "flee". "Attach" never means "flee", so the term

"bolt" has an ambiguous meaning. Resolving this uncertainty is called sense disambiguation, and the task of discovering senses is called sense induction.

This hypothesis is especially relevant when using syntax-based approaches to semantics. Syntactic behavior is likely to be different for these different meanings, Ignoring polysemy introduces noise, which causes two problems for cluster recovery. A polysemous verb treated as a single unit has many syntactic frames, including frames from different semantic classes, which makes it align poorly with any well-constructed classes of syntactic patterns. Further, once that polysemous unit is assigned to a cluster, it degrades the coherence of the cluster by introducing syntactic frames which do not belong.

The importance of capturing polysemy is highlighted in prior work on verb clustering (Korhonen et al., 2003).

Hypothesis 3 *Sense disambiguation can be done simultaneously with semantic cluster creation.*

The baseline model, introduced in Chapter 3, is a two-step framework for creating semantic verb clusters, separating the learning of verb senses from the learning of verb clusters. The sense induction model effectively works by building semantic groups of co-occurring arguments to the verb, like topics from latent Dirichlet allocation (LDA) (Blei et al., 2003), using a separate set of topics for each verb. Then these verb senses are clustered across verbs, using a set of syntax features as topics to describe the syntactic behavior of verb clusters. This model requires processing the corpus twice, and does not share any information across verbs.

The joint model, also introduced in Chapter 3, provides sense disambiguation and clustering in a single step, and offers several advantages. The learned senses are actually better than the step-wise system, and the clusters can be improved by incorporating the full set of features from both steps of the baseline model. Because there is no layer of abstraction between sentences and clusters, sentences with VerbNet class labels can be used to help guide

the cluster creation (Chapter 5), providing further benefit. Further, the model is conceptually and mathematically more similar to LDA, so can be implemented easily by making minor modifications to existing codebases.

Hypothesis 4 *Partial supervision can increase both the accuracy and the coherence of automatically-created semantic verb clusters, even for clusters with no supervised examples.*

Machine learning is about teaching computers to uncover and recognize patterns; when the patterns are a set of known labels for some examples, it is generally called supervision. Preliminary experiments adding supervision to Bayesian verb clustering, described in Chapter 5, indicate that VerbNet alignment can be improved by adding supervision. Overwhelming evidence from the body of machine learning research supports the notion that supervision will help classification on the provided classes; this hypothesis supports a stronger notion that supervision need not describe all behavior of interest in order to improve the quality of all clusters. Evidence provided in Chapter 5 demonstrates that the accuracy of the clusters is improved even if a cluster is not included in the supervision, but the coherence judgements are still inconclusive. There are open questions about the best way to add partial supervision to clustering techniques, and best practices seem to depend on the domain, so this is a promising line of future research discussed in Chapter 6.

1.2 Publications

Kawahara et al. (2014a) introduced a model for verb sense induction using probabilistic models, which is covered in Section 3.2. Later, Kawahara et al. (2014b) clustered these induced senses into polysemy-aware verb clusters that aligned well to VerbNet, creating a step-wise framework covered in Section 3.3.

Peterson et al. (2016) was published at *SEM 2016, and demonstrates a method to improve the step-wise framework by including distant supervision from SemLink. The methodology and results are covered in detail in Section 5.1.

Peterson and Palmer (2018) was published at AAAI 2018, and includes several contributions that improved the step-wise framework. First, it introduced a model that performs sense induction and semantic clustering for all verbs simultaneously. This model is nearly identical to latent Dirichlet allocation (LDA) (Blei et al., 2003), treating each verb as a document. Second, it introduced a novel model with a unique mathematical basis to improve the verb clustering step. Peterson and Palmer (2019) is currently under review, and is a short paper discussing the limitations of this novel model. These results are covered in Sections 3.4.1 and Chapter 4.

Peterson et al. (2019) is currently under review, and introduces a method for partially supervising the joint sense induction and clustering model (Peterson and Palmer, 2018). The semi-supervised clustering method is extremely computationally efficient, requiring no further changes to the inference algorithm. The resulting clusters align to VerbNet more closely than any previous model, including the method of adding supervision to the step-wise framework (Peterson et al., 2016). These results are detailed in Sections 3.4.2 and 5.2.

During this research, we carried out several experiments on supervised topic modeling at Oracle Labs. Oracle has filed a provisional patent covering this work, which includes, in part, the computationally-efficient method of partial supervision described in Section 5.2 (Peterson et al., 2019).

1.3 Outline

Chapter 2 defines important terms and describes the foundation of ideas this work is built upon. Chapter 3 describes the Dirichlet-multinomial and Dirichlet process mixture models that form the scaffold of the experiments in this research. Chapter 4 describes a novel clustering model based on the positive pointwise mutual information (PPMI), and illustrates its success and limitations. Chapter 5 describes two methods of adding supervision

from Semlink, one for the step-wise model and one for the joint sense induction and clustering model. Chapter 6 ties this work together, and provides some insight into future directions for this research.

Chapter 2

Background

This work, like all human accomplishments, relies on a foundation of established knowledge. It employs theory from both linguistics and machine learning. In order to place it in proper context, we must first understand the linguistic principles that serve as inspiration, the resources used for evaluation, and the corpora that provide the required input knowledge. We will then discuss probability theory and how probabilistic models can be used to model semantics. This work builds semantic clusters, so we will discuss the general case of clustering, define polysemy and cast in in terms of clustering, and review the history of creating semantic verb clusters. Finally, we'll cover a history of capturing meaning using vectors, which admit easy representation and manipulation by computers.

2.1 Corpora and Parsing

Language is a basic human faculty. It allows us to transmit ideas, with the basic assumption that our ideas can be understood by others. However, it is not obvious how this works. We want to learn more about how language works because it has practical and philosophical importance.

The scientific process, in which ideas are held up to examination by experiment, is the best way to gain knowledge about the world. Scientific examination of language is called

linguistics, but in order to test our ideas we must have a testing apparatus. Linguists have developed many methods to gather experimental evidence, but in this work we primarily use corpora.

A corpus (plural corpora) is simply a body of text, composed of many distinct works. A collection of newspaper articles, or medical abstracts, or works of fiction, is a corpus. In some sense, an encyclopedia or even a full library may be a corpus.

Corpus linguistics employs observations of real language use as evidence to make inferences. Each document in a corpus, and each sentence in the document, was written by an author trying to convey an idea. If a corpus contains a wide range of authors discussing a wide range of topics, it should actually reflect a fairly complete range of accepted language use. Of course, many terms are used only in particular domains ("contraindicated" is seldom used outside of medical contexts, and "abet" is seldom used outside criminal law), and almost any corpus will be missing terms that exist in others. In a large enough corpus, even uncommon terms will typically occur with high enough frequency to reason about their typical patterns of use. Corpora are essentially treated as true linguistic evidence, and we can use this observational data to validate hypotheses about how language works.

Further, annotated corpora provide an important function for training machine learning systems to perform linguistic tasks. In natural language processing the goal is often to train a computer to extract meaning, to translate, summarize, or find the structure of a sentence. By treating a corpus as a set of example sentences, and explicitly labeling the structure that should be derived for each sentence in the corpus ("annotating"), we guarantee that the machine learning system has a training set of examples that look like the real data it may encounter. If the annotated corpus is sufficiently large, it will contain varied examples, and can train a sophisticated and accurate recognizer for the desired behavior.

In particular, this work uses corpora that are completely digital, and contain primarily modern English. Working with digital corpora makes it easier to start analysis using computers, and is hardly restrictive since the vast majority of text produced in the last

three decades has been digital in the first place. English is one of the major languages for international communication, and as a result has many large and widely-available corpora. Standard and often-used corpora for computational linguistics include dumps of Wikipedia (wik), or the large Gigaword corpus (Parker et al., 2011) (distributed by the Linguistic Data Consortium).

2.1.1 Parsing

Parsing is the act of converting a sentence into a structured syntax tree Jurafsky and Martin (2014); Manning et al. (1999). It is a basic task in computational linguistics, and one of the prerequisite steps for this work. Tbere are several processing steps involved to produce a parse of a sentence, and each has been given attention as a task in its own right.

The main preprocessing steps are sentence segmentation, tokenization, and part-of-speech tagging (Jurafsky and Martin, 2014). The raw text must be broken into sentence units, and while the boundaries are typically marked with punctuation, punctuation marks aren't perfect indicators (e.g., abbreviations are often marked with periods, question marks may appear inside quotes). Each sentence must have its words separated into "tokens," which are the raw units of text. Each word is a token, but a word with a trailing comma should be treated as two tokens: the word, and the comma. For this reason, punctuation marks are treated as independent tokens, again with some exceptions (e.g., apostrophes and the aforementioned abbreviation periods). The rules for tokenization are thus a little complex, but generally agreed upon[1] Finally, each token is labeled with a "part of speech" - these are the typical word types (noun, verb, adjective, adverb) and also some specific special cases of linguistic relevance (punctuation, proper noun, separation of singular from plural in nouns and gerund from infinitive in verbs). These preprocessing steps are standard input to many natural language processing tasks, and there are high-quality, free systems available

[1]Tokenization rules are much more complicated in Chinese than in English, since Chinese words may span multiple characters, and deciding which characters belong together as a unit is nontrivial. English has a lovely tendency to split words with spaces, but this isn't a universal, and this splitting completely ignores multi-word expressions (e.g., "heart attack") that may be better to treat as a single token.

to perfom them (Bethard et al., 2014; Bird et al., 2009; Manning et al., 2014).

Parsing, or converting a sentence into a syntax tree, can be done in two ways, called "constituency" and "dependency" parsing. This work uses dependency parses to represent the syntactic information in a corpus, but both types of parsers are briefly described.

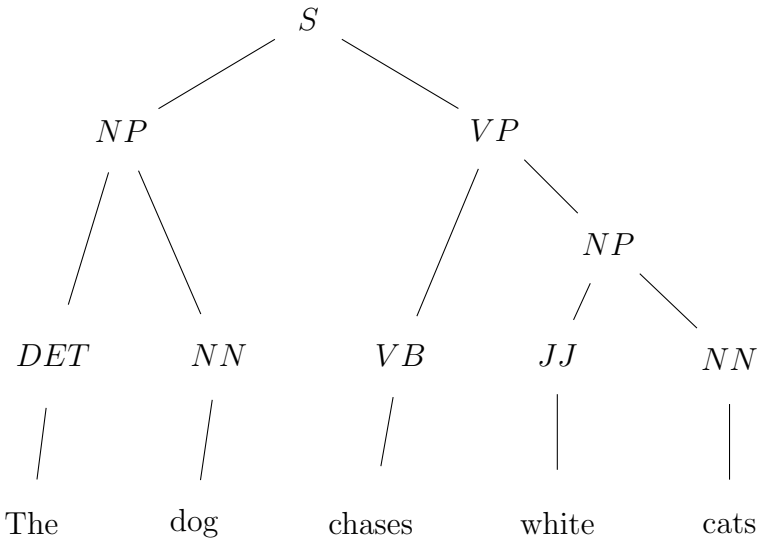

Figure 2.1: An example constituency parse.

Constituency parsing breaks a sentence (or, recursively, a phrase) into its syntactically-coherent constituent parts. Figure 2.1 shows a constituency parse for the sentence "The dog chases white cats." The "leaves", or terminal nodes of the tree, have a one-to-one correspondence with the tokens of the text. Each non-terminal node is labeled with a summary of the "type" of phrase it represents. Some types are coarse, such as "noun phrase" (NP), which contains a noun along with any determiners or modifiers that appear in conjunction; others are essentially parts of speech, such as determiner (DET) or adjective (JJ). Each constituent, regardless of granularity, has a unified grammatical role in the sentence, and that role is labeled. The example sentence can be broken into two coarse constituents: a noun phrase ("the dog"; NP) and a verb phrase ("chases white cats"; VP). The verb phrase itself has two finer-grained constituents: a verb ("chases"; VB) and a noun phrase ("white cats").

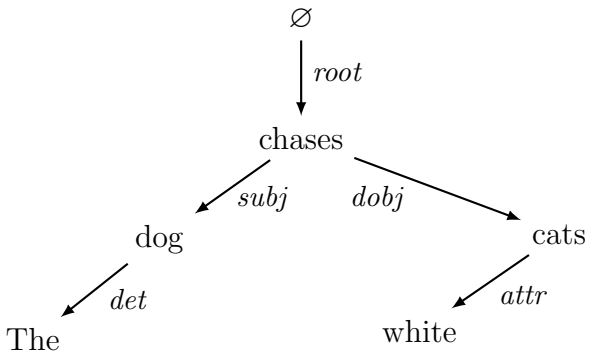

Figure 2.2: An example dependency parse.

Dependency parsing attaches words together with directed links. Figure 2.2 shows a dependency parse of the same example sentence. The main verb of a sentence ("chases" in the example above) is linked to an always-present "root" node, that is named for being the root of the tree. The dependents of the verb, which may include subjects, direct and indirect objects, clausal complements, and other syntactic structures, are linked with paths leading from the verb. However, if the dependent is a multi-word phrase, only the "head" word is a dependent of the verb. In the above example, "The dog" is the subject of "chases", but "dog" is the head word of the phrase. "The" depends on "dog", not "chases". Similarly, "white" depends on "cats." Each word in the sentence is the direct descendant of a word, and may be the head word of its own syntactic sub-tree.

An important aspect of dependency parsing is labeling the links. Typically, the subject of a verb is labeled differently from the object, because these are distinct syntactic relationships. Because each link goes to exactly one word, the main syntactic structure of the sentence can be read from the direct dependents of the verb, and the relationships are clear.

This work uses dependency parses, which are general and compact. Although some nuance is lost, "dog chases cats" is the syntactic and semantic core of the example sentence above. It helps to use collapsed prepositional dependencies, so if the prepositional phrase "on Tuesday" was linked to a verb, the head word "Tuesday" would be linked directly to the verb, with label `prep_on`. With these collapsed dependencies, all the most relevant information to

the verb and its context is preserved in one single link, and everything lower in the tree can be removed.

There are corpora that have been annotated with parse information (Silveira et al., 2014). Though labeling such a corpus required a large investment of resources to create, there are now enough example parses to successfully train algorithmic parsers (Chen and Manning, 2014), and achieve high-quality output. These parsers can be used to label orders of magnitude more data than could be feasibly accomplished by humans, and do so quickly. The dependency relations analyzed in this work are all automatically generated by algorithmic parsers.

2.2 Semantics: Models of Meaning

Semantics is the study of meaning. One useful approximation is describing *who* did *what* to *whom*, *when*, *how*, *where*, and *why*.

In Section 2.2.1, I will describe semantic role labeling, which is a widely-used framework for labeling the parts of a sentence that answer the questions above. Section 2.2.2 describes semantic resources, including PropBank, WordNet and FrameNet. Because this work deals extensively with VerbNet, Section 2.2.3 describes this particular resource in more detail. Section 2.2.4 describes SemLink, which includes a unified annotated corpora with VerbNet, PropBank, and FrameNet labels.

2.2.1 Semantic Roles

Semantic role labeling (SRL) (Palmer et al., 2010) is the task of labeling the semantic participants (predicates and arguments) in a sentence. Each argument is labeled with its "role," which indicates its relationship to the predicate. Consider the sentence, "Bob broke the window." Here the predicate, or action, is "break," and it's the *what* of the sentence. The "agent" is the argument that performs the action, "Bob" (*who*). The "patient" is the

argument that is acted upon, "window" (*whom*). Linguists use the role "patient" especially to describe arguments that are changed by the action; if Bob only looked through the window, the window should instead be the "theme." If he broke the window with a hammer, the hammer is the "instrument" of the breaking, and if he gave the hammer to Alice, then she is the "recipient." Of course, not every sentence has an agent; sometimes, "The window simply broke." Here, although window is in the syntactic subject position, it is still the patient of the predicate.

PropBank is a corpus labeled with verbal, nominal, and adjectival propositions and their arguments, and it defines semantic roles for each predicating element. The underlying text for PropBank is the same as the Penn Treebank, so the semantic role labels are an additional layer of annotation, on top of the parse trees. PropBank defines numbered semantic roles for each verb, and are somewhat coarse- grained. Typically, PropBank uses `arg0` to denote the prototypical agent, and `arg1` to denote the prototypical patient. The notion of "prototypical" agency was introduced by Dowty (1991), who argued that role types should be viewed not as discrete types, but as prototypes with a list of verbal entailments. For example, if "The ball broke the window," the ball is still the `arg0` because it fits the "breaker" proto-role, even though it doesn't have any particular agency in the action. Similarly, `arg1` may be used for themes, or experiencers, for some verbs. Numbered arguments can also denote verb-specific additional roles: for "buy", `arg0` is the buyer; `arg1` is the thing bought; `arg2` is the seller; `arg3` is the price paid; and `arg4` is the beneficiary. Obviously, any particular sentence may contain a subset of these arguments. Not all roles are obligatory.

Semantic role labeling is often treated as an intermediate task, and supports systems like information extraction and question answering (Shen and Lapata, 2007; Christensen et al., 2010; Moreda et al., 2011).

2.2.2 Semantic Resources

There are several noteworthy semantic resources aside from PropBank.

WordNet (Miller, 1995) acts as a dictionary with explicit links that capture word-level semantics. "Synsets" are groups of words that are synonymous (mutually interchangeable), and are labeled and linked to other synsets with relations like antonymy (having opposite meaning, as "bad" is to "good"), hyponymy (one is a kind of the other, as a car is a type of vehicle), and meronymy (one is a part of another, as a tire is a part of a car). Each synset also has a part-of-speech label, so the verb "bank" is in a separate synset from the noun. Polysemy (where the same word may have multiple meanings, like "bank" as a noun may be either a financial institution or the side of a river) is accounted for by numbering the distinct senses, and putting each into a synset independently.

FrameNet (Baker et al., 1998) collects the semantic "frames" of various verbs, and is a more abstract representation. A semantic frame is a description of an event, which may be invoked by a predicate. For example, "commerce_buy" is a semantic frame that may be invoked by a predicate like "buy" or "purchase", but shares certain semantic references. As in PropBank, Framenet lists the semantic participants (a buyer, and the goods purchased, and optionally a seller, money exchanged, a beneficiary, or an explanation), but lists words that can invoke the frame (like a WordNet synset), and its relationship to other frames (inheritance, frames it relies on or that rely on it. For example, "import" invokes a frame that relies on "commerce_buy"). This mapping of relationships describes the "network" of semantic frames.

2.2.3 VerbNet

VerbNet (Kipper-Schuler, 2005) is a semantic resource for verbs. Like WordNet and FrameNet, it links words that are semantically similar. Unlike other resources, VerbNet is a verb-specific lexicon, and each VerbNet "class" (cluster of semantically-related verbs) is described with thematic roles (similar to FrameNet, but typically less specific), the restrictions on the arguments (e.g. whether the subject must be a person, or must be animate), and a set of syntactic frames. In VerbNet, a frame is a syntactic realization (e.g. transitive,

intransitive, or ditransitive sentences, whether prepositional attachments are allowed, and whether a sentential complement is allowed or expected), and an associated set of semantic predicates, which provide general semantics of the event described by the verb.

VerbNet is based on the classes of Levin (Levin, 1993). Levin's classes are organized by the hypothesis that semantics has a strong impact on syntax, and therefore that syntactic distributions contain useful semantic information. Verbs share a class when they can appear in the same set of syntactic frames, and share a set of semantic predicates. This hypothesis, which we call Hypothesis 1, has so far held up to linguistic scrutiny. VerbNet extends the coverage of these initial classes to a much larger set of predicates. VerbNet also extends the structure hierarchically, allowing sub-clusters that share specific behaviors to be distinguished from more general classes.

VerbNet's semantic information is extremely rich, and is designed to aid semantic processing. Any sentence with a VerbNet class label has instant access to a list of semantically similar verbs, selectional restrictions on the observed syntactic frame, thematic roles, alternate allowable syntactic frames using those same roles, and a set semantic predicates that describe the general event semantics of the sentence. These pieces may be used in automatic translation, for example, to generate simpler paraphrases of a sentence which may be easier to translate into the target language. Or, when tracking events through a news story, they may be used to reason about whether any of the semantic participants in a sentence has changed state.

VerbNet class annotations have proven useful semantic role labeling (Giuglea and Moschitti, 2006; Hartmann et al., 2016). Its success in supporting NLP tasks has led to the creation of similar resources in other languages, such as Urdu (Hautli-Janisz et al., 2015), French (Pradet et al., 2014), Basque (Aldezabal et al., 2010) and Arabic (Mousser, 2010).

The main drawback of VerbNet is its lack of coverage. VerbNet is in English, and similar resources exist in only a handful of languages, even though it may be especially useful for processing in low-resource languages. Further, VerbNet's coverage in English has gaps, and

offers limited support for restricted domains (such as legal or medical texts). Adding coverage to VerbNet is difficult, and the task grows more difficult as the resource grows; making the correct classification is harder when there are more options, and some necessary classes are likely to be missing. However, the semantic information of VerbNet is especially necessary when dealing with rare words because these words have fewer examples to learn patterns from. Expanding VerbNet's coverage manually for all possible domains is an overwhelming, likely infeasible task. Developing VerbNets for other languages is even more challenging.

Automated approaches to generating VerbNet-like structures are promising because they solve the coverage issue directly. A resource that builds verb clusters based on a corpus will have coverage of that same corpus. The challenge lies in aligning automatic verb clusters to VerbNet in order to gain the semantic information associated with VerbNet classes. In this work I aim to improve automated approaches in two ways. First, I aim to tease out the necessary components of making VerbNet-like structures based only on corpus counts (Hypotheses 2 and 3). These models are more generalizable, and portable to new domains and languages much more easily. Second, I aim to improve alignment using limited labeled data (Hypothesis 4), because high-accuracy alignment with VerbNet is required in order to gain the benefits of VerbNet's semantic annotation. Chapter 5 shows that each of these investigations can improve VerbNet alignment, and presents a new state of the art model that incorporates all the improvements from this work.

2.2.4 SemLink

Semlink (Palmer, 2009; Bonial et al., 2013b) is a project that links together the semantic resources PropBank, WordNet, FrameNet, and VerbNet. Semlink version 1.1.2 contains annotations of roughly 78,000 tokens from the Wall Street Journal section of the Penn Treebank corpus. These annotations contain the semantic roles, VerbNet classes, and FrameNet frames associated with each verb. It is useful for training tools that allow automatic linking of these resources, but is also a useful corpus of VerbNet class annotations. We use it

extensively in this work, both as an evaluation set and later as a source of supervision.

2.2.5 Recap of Semantics Background

Section 2.2 introduced the basic notions of semantics and semantic role labeling (2.2.1). It also described useful semantic resources such as PropBank (2.2.1), WordNet and FrameNet (2.2.2), VerbNet (2.2.3), and SemLink (2.2.4). It also defined polysemy, which will be a recurring theme throughout this work. Polysemous words have multiple meanings, but the same spelling. These meanings may not be related in any obvious way, so polysemy is an important phenomenon to characterize language use.

2.3 Bayes' Theorem

A great deal of this work deals with probabilities. Section 2.3.1 provides a primer on very basic probability, and Section 2.3.2 introduces a few common probability distributions, and the notion of expected value. Section 2.3.3 describes probabilistic graphical models, which model the world by stating assumptions about how observed variables are related. These models find explanations of data that balance the tradeoff between accuracy and the strength of the assumptions, and often learn good representations. One extremely popular graphical model for textual analysis is latent Dirichlet allocation (LDA), which is described in Section 2.3.4. Section 2.3.5 describes supervised topic modeling, which is a family of techniques that incorporates known document labels as part of the graphical model. Incorporating these known labels can steer the model to find representations that describe the data, while simultaneously optimizing that representation to learn specific patterns. Finally, Dirchlet processes (Section 2.3.6) can be used to sidestep one major challenge in LDA, which is choosing a good number of clusters. Dirichlet processes instead sample the number of clusters, with a strong assumption that the number of clusters should be small.

2.3.1 Basic Probability

We encounter probability all the time in daily life. All outcomes are uncertain when flipping coins, rolling dice, or shuffling cards. We also know, on some level, that every time we get in a car, we may blow a tire on a stray nail, or that a lottery ticket may win us millions of dollars. What is amazing about studying probability is that it gives us the tools to precisely describe the chances of any outcome. The sum of two dice is three times as likely to be 7 as it is to be 3; the odds of a flat tire are low but easily calculable for a trip of known length; and the odds of a Powerball ticket winning the jackpot are lower still.

Probability theory is based upon two very simple assumptions (mathematically, these are called axioms). The first is that we will reflect probabilities as being odds, where an event that will surely happen has probability 1, and an event that will never happen has probability 0. The second is that no event has negative probability.

Consider a well-shuffled deck of playing cards. Trivially, we can observe there is a one in four chance the first card is a club ($P(C) = 1/4$), since there are four suits. There are thirteen ranks, so there is also a one in thirteen chance it is an ace ($P(A) = 1/13$), and the same chance it is a 4 ($P(4) = 1/13$). It can be both an ace and a club, but and there is exactly one such card in a deck of 52 ($P(AC) = 1/52$). Because $P(AC) = P(A) * P(C)$, we call these events independent. It cannot be both an ace and a 4, so $P(A4) = 0$. But, because it cannot be both, we can compute the probability of it being an ace *or* a 4 by simply summing the individual probabilities ($P(A \text{ or } 4) = P(A) + P(4) = 2/13$). Because the first card will surely be a club *or* not a club, so $P(C \text{ or } \neg C) = 1$, and since it cannot be both, $1 = P(\neg C) + P(C)$.

If I observe the first card before you see it, and give you information about it, it should affect your assumptions of the likelihood of the card. If you know it is a club, you know there is a zero chance it is a spade. Mathematically, we denote "probability of S given C" as $P(S|C) = 0$. We also know $P(C|C) = 1$ and $P(A|C) = 1/13$. In general, we can compute

the conditional probability for events A and B as

$$P(A|B) = \frac{P(AB)}{P(B)}. \tag{2.1}$$

This makes sense, because the probability of A, knowing B, must be related to the probability of them occurring together. And, if B is known, we don't have to consider any part of the full distribution where B doesn't occur, so in order to make $P(B|B) = 1$ as we would expect, we need to divide by $P(B)$.

Conditional probability gives further insight to the notion of independence. Recall A and B are independent when $P(AB) = P(A) * P(B)$. Here, $P(A|B) = P(AB)/P(B) = P(A) * P(B)/P(B) = P(A)$, and similarly $P(B|A) = P(B)$. So, for independent events, knowledge of one doesn't change the probability of the other.

Bayes' theorem builds on the relationship between joint probability and conditional probability. Consider Equation 2.1. We may restate this as,

$$P(AB) = P(A|B) * P(B). \tag{2.2}$$

We could also show, from $P(B|A)$, that

$$P(AB) = P(B|A) * P(A). \tag{2.3}$$

Since $P(AB)$ is the same in both Equations 2.2 and 2.3, the right hand sides must also be equal, and with one simple algebraic step,

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}. \tag{2.4}$$

This seems simple enough an idea, but it is incredibly powerful. It lets us compute the probability of $P(A|B)$ without needing to compute $P(AB)$, which often depends on knowledge

of the full joint distribution. For a deck of cards this is simple, but we are often interested in probabilities with billions of possible events in the joint distribtuion, and computing $P(AB)$ is computationally infeasible, even with a supercomputer. Computing the conditional probability of $P(B|A)$ and the probability of a single event $P(A)$ or $P(B)$ is much easier, and much faster.

The most important properties of probability theory are as follows.

- For any event A, $P(A) \geq 0$.
- For any event A, $P(A) + P(\neg A) = 1$.
- For disjoint events A and B, $P(A \text{ or } B) = P(A) + P(B)$ and $P(AB) = 0$.
- Events A and B are independent if $P(AB) = P(A) * P(B)$.
- The probability of A when B is known is $P(A|B) = P(AB)/P(B)$.
- The above can be expressed as $P(A|B) = P(B|A) * P(A)/P(B)$.

2.3.2 Distributions and Probabilistic Functions

A probability function with mutually disjoint outcomes is called a "distribution" because it describes how the fixed probability mass (recall, the sum of all disjoint events is 1) is distributed to each outcome. A distribution implies a sort of scenario: drawing cards from a deck, or measuring the typical waiting time between severe storms, or predicting the price of a stock in the future. Some general scenarios occur commonly, and these probability functions are studied and named as standard distributions. I will focus on characterizing the distributions that are relevant to this work.

The uniform distribution, $\mathbb{U}(X)$, characterizes a draw of a random, real number x from the interval $[0, X)$. In particular, it is called uniform if every real number in that interval has an equal chance of being drawn.

The multinomial distribution, $Mult(\theta)$, characterizes a roll of a weighted die, with a fixed number of sides. θ is a vector of non-negative real numbers, whose sum is 1; each component θ_i describes a probability for a particular outcome i. When we roll the die, we show value i with probability $P(i) = \theta_i$.

A variable x drawn from a distribution is called a random variable, since we do not know its value until performing the draw. We use the notation $x \sim \mathbb{U}(X)$ to denote x being drawn from a uniform distribution with maximum value $X > 0$.

The "expected value" $\mathbb{E}(x)$ for a random value x is essentially the mean of the distribution from which x is drawn, weighted by the chance of choosing it. For a continuous distribution like the uniform distribution, we can compute the expected value using $\mathbb{E}(x) = \int_{-\infty}^{\infty} xP(x)dx$. For a discrete random variable drawn from the multinomial distribution, we compute $\mathbb{E}(x) = \sum_i iP(i)$.

The Dirichlet distribution, $Dir(\alpha)$, is a distribution that takes a vector α of concentration parameters. A draw from a Dirichlet $\theta \sim Dir(\alpha)$ is a vector of the same length as α, having the properties that all elements of θ are nonzero, and $\sum_i \theta_i = 1$. That is to say, a draw from a Dirichlet distribution may be treated as the parameters of a Multinomial distribution. The vector θ, on expectation, will be greater where α is greater, and the larger the components of α, the smaller the variance that will be allowed from that mean.

2.3.3 Probabilistic Graphical Models

In this work, we will use the term model to mean a mathematical description of the world. The model describes the structure that we expect to see in the world, and we "fit" a model by finding parameters that make the model match our observations. Once a model has been fit, it can exploit its structure to reason about the world it has not observed. It is useful to the extent that it makes correct predictions about new data. When we talk about the accuracy of a model we always mean that we fit the model to some data and evaluate its accuracy on some intentionally held-out, extra data.

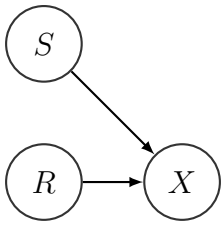

Figure 2.3: A graphical model describing the relationship of the first card drawn from a deck, X, to the suit, S, and rank R. S and R are not linked, because they are independent variables.

In this work, we particularly use the term model to mean a probabilistic graphical model. We call the models graphical because we build a graph describing which variables are related to one another, and what the relationships are. Each relationship is probabilistic, encoding conditional likelihood once the parent variable is known.

We again use the well-shuffled deck of cards to illustrate a simple probabilistic graphical model. The first card, X, depends on both the suit, S, and the rank, R. There is no relationship between the suit and the rank, because they are independent. Knowing S influences our knowledge about X, so there is clearly a relationship between these two variables. A similar relationship holds between R and X. Figure 2.3 draws the variables and shows their relationships.

2.3.4 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is a graphical model that gives a generative story to the creation of a corpus of text. Essentially, it posits that each document in a corpus is a bag of words drawn from some hidden (or "latent") distributions. The document has a distribution over words, but that distribution is created from a mixture of the latent distributions[2], which are often called "topics." These hidden topic distributions are drawn from a Dirichlet distribution, which is chosen so that the multinomials tend to be

[2]In fact, LDA is in a category of models called Bayesian mixture models. The observed data are drawn from a mixture of sources, and generally mixture models try to learn what those sources are, subject to prior knowledge that put constraints on likely mixture components. These are models that have been very effective at modeling a huge variety of phenomena as clustering algorithms.

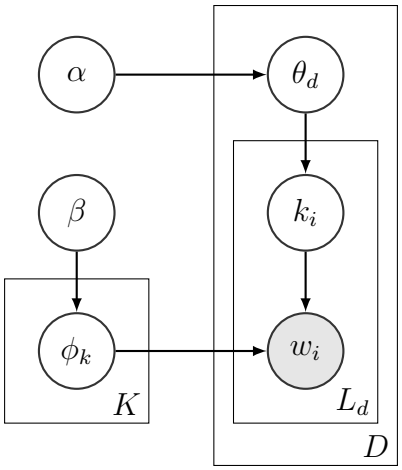

Figure 2.4: The generative model for latent Dirichlet allocation. Square plates indicate repetition, and are notated with the total number of times the plate is repeated in the lower right corner. Each of the D documents has a unique distribution over topic, θ_d, drawn from a Dirichlet distribution with parameter α. Each of K topics ϕ_k is drawn from a Dirichlet distribution with parameter β. Inside a document, L_d words are drawn by first selecting a topic k_i from θ_d, and then selecting w_i from the corresponding ϕ_{k_i}. In practice, w_i are the observed tokens in a corpus, and so this node is shaded.

highly focused on a relatively small number of words. Also, the document's mixture of topics is itself a multinomial distribution, drawn from a separate Dirichlet distribution. Again, the Dirichlet prior encourages the document to be a mixture of a small number of topics. In the generative model (Algorithm 1), documents are written, word by word, by sampling a topic at random from the document's distribution, and then drawing a word at random from the corresponding topic distribution.

Although the generative process is not an accurate description of how a document gets written, it encourages useful topics. Each topic is focused on a small set of keywords, so it tends to be coherent. Each document is focused on a small number of topics, so words that occur together are naturally pulled into the same topics. The latent topic distributions are characterizations of the coherent topics that are shared across all the documents, and accurately model the documents and terms that are actually observed.

The graphical model for LDA is shown in Figure 2.4.

Algorithm 1 Generating a Corpus using Dirichlet-Multinomial Mixtures

Select a number of topics, K
Select a number of documents, D
Select concentration parameters, α and β
for each topic $k \in [1, \ldots, K]$ **do**
    Draw $\phi_k \sim Dir(\beta)$
end for
for each document $d \in [1, \ldots, D]$ **do**
    Select a document length, L_d
    Draw $\theta_d \sim Dir(\alpha)$
    for word $i \in [1, \ldots, L]$ **do**
        Draw topic $k_i \sim Mult(\theta_d)$
        Draw word $w_i \sim Mult(\phi_{k_i})$
    end for
end for

Training in LDA

LDA describes a probabilistic semantic decomposition of a corpus, but in order to use it, we must find the parameters (topics and document-topic distributions) that allow the generative model to most-closely match the corpus. This is called training, and broadly may be accomplished in two ways.

Variational inference is one method of training graphical models that sacrifices some of the expressive power in order to have faster training. It is similar to the training of Word2Vec in that we define an objective function for which we can compute the gradient, and seek an optimum solution numerically.

Sampling techniques, broadly called "Markov chain Monte Carlo" (MCMC), are used in this work over variational inference. They are exact and easy to implement. In general, they tend to be slower, but practically speaking they are adequately fast. They can take advantage of multiple processors and be distributed across clusters of machines (Zaheer et al., 2016), which helps dramatically, and the mathematical operations are much simpler. Because they are simple to implement, they are much easier to modify. Adding supervision and other complications to a variational algorithm is a much more demanding task.

In particular, LDA is often trained using Gibbs sampling, an MCMC technique that

updates one variable at a time. Even starting from random topics, updating the assignment of a single word will tend to improve the distribution of topics. The Dirichlet prior on the multinomials encourages the multinomials to put weight on only a small number of outcomes. In sampling, this manifests through a "rich-get-richer" effect. We update the topic assignments of all the words in the corpus, one after another, and each time we are more likely to use the topics we've used often in that document, and that already place a high chance on the observed word. As we continue to re-sample words based on these updated topics, the rich-get-richer effect becomes more pronounced, and we end up with an assignment that is dramatically better at using only a few topics per document, and a few key terms per topic. This technique doesn't require us to compute the gradient of the full objective. Instead, we compute the statistics one variable at a time. Mathematically, it is guaranteed that after enough passes through the data, the topics do not depend on the initial topic distributions.

Gibbs sampling in LDA is described in Algorithm 2. It has a simple mathematical form, because of a unique property of the Dirichlet distribution. The Dirichlet is a conjugate prior of the Multinomial distribution. This is a precise mathematical term, but practically speaking, it means that if we have:

- a Dirichlet distribution $Dir(\alpha)$;
- a Multinomial distribution $Mult(\theta)$ such that $\theta \sim Dir(\alpha)$; and
- observations of draws $x_1, x_2, \ldots, x_n \sim Mult(\theta)$;

then we may compute $P(x \sim Mult(\theta) = k)$ without ever needing to know θ explicitly[3]. In particular, if $C(k)$ is the count of times k appeared in our list of samples $x_1, x_2, \ldots, x_n$, then

$$P(x = k) = \frac{C(k) + \alpha_k}{\sum_i C(i) + \alpha_i} \tag{2.5}$$

[3]Actually, we can integrate over all possible distributions θ, each weighted by their likelihood. This allows us to skip sampling any particular θ.

Throughout this work, I will often use $*$ to stand in for an aggregate, i.e. $C(*) = \sum_i C(i)$. Also, note the denominator does not depend on k at all. When it seems more legible, I will instead write the above equation as

$$P(x = k) \propto C(k) + \alpha_k,$$

where the $\propto$ implies there is a normalization step required to make the result a proper probability, that is omitted in the equation. These are typical conventions in the literature, because they help emphasize the details necessary for understanding the equations. Further, I will refer to this sampling where a Multinomial is drawn from a Dirichlet, as a Dirichlet-Multinomial.

When looking at the generative process, each word is generated by first selecting a topic k from the the document's distribution over topics, θ. Then, a word is drawn from that topic's distribution over vocabulary items, ϕ_k. The words are actually our observed variables, and the unknown variables are the topics that were chosen at each step. In fact, we can view fitting as learning the topic assignments, one per word, that were used in the assumed generative process. We start with a totally random assignment to topics, and we treat each assignment as an observation drawn from a Dirichlet-Multinomial. We know that the topic should be chosen from the document, and we have observations of topic assignments for the other words: by Equation 2.10, we can compute this simply using these counts. Also, we have the set of all words assigned to each topic in the corpus; these observations are sufficient to compute the probability of any given word from the corresponding topic's Dirichlet-Multinomial.

That is, we can observe counts $C_\theta(k)$ of topic assignment k inside the document's θ. This allows us to compute $P(k|\theta) \propto C_\theta(k) + \alpha_k$. We also know the counts $C_{\phi_k}(w)$, allowing us to

compute $P(w|\theta_k) \propto C_{\phi_k}(w) + \beta$. With Bayes's rule, we see

$$P(k|\theta, \Phi) \propto (C_\theta(k) + \alpha_k) \frac{C_{\phi_k}(w) + \beta}{C_{\phi_k}(*) + |V|\beta} \tag{2.6}$$

Algorithm 2 Gibbs sampling in LDA

```
Assign each word to a random topic and compute counts matrices
for iteration in range(num_iters) do
    for document d in corpus do
        for word w, with current assignment k, in d do
            Decrement counts of observations C_{θ_d}(k) and C_{φ_k}(w)
            Sample topic k_new according to Equation (2.11)
            Increment counts of observations C_{θ_d}(k_new) and C_{φ_{k_new}}(w)
        end for
    end for
end for
```

2.3.5 Supervised Topic Modeling

One nice feature of graphical models is that it is easy to extend them. Since a graphical model is a description of how variables are interrelated, new variables can be introduced, as long as their relationship to other variables is understood.

Supervised LDA (Mcauliffe and Blei, 2008) and DiscLDA (Lacoste-Julien et al., 2009) both add a secondary classification task to the training objective, so that the topics assigned to each topic are effective features to classify the document according to a fixed label. Labeled LDA (Ramage et al., 2009) designates a single, known label to each topic, and allows documents to have multiple labels. The labels on the documents provide hard constraints on the available topics, so the learned topics conform to the label set, regardless of semantic interpretability.

A simple case involves document classification. Consider building a topic model that characterizes a corpus of emails, but also classifies each message as belonging to a category: "business," "personal," or "spam." We first generate a topic distribution for each document, and then from that distribution we generate the label, along with a set of words. When

sampling, the topics will still tend to put words in the same document together. However, a topic is more likely to be shared by two emails with the same label, and less likely if the labels are different. In sampling, the model will try to find topics that are concise, coherent descriptors of the corpus, but also are good at separating the known document classes. The input labels are the supervision.

These techniques require accurate and complete document labels in order to be effective, and limit the applicability to semi-supervised domains. They aren't really suitable for our task, because we know some verbs have senses not currently catalogued in VerbNet, and because many verbs are missing entirely, so we do not have an exhaustive labeling of our documents.

There are several methods of including word co-occcurrence knowledge or constraints to help ensure topics conform to user-specified constraints (Xie et al., 2015; Yang et al., 2015; Hu et al., 2014; Andrzejewski et al., 2009; Jagarlamudi et al., 2012), that allow users to specify words that must or must not belong together, and in doing so guide the output of the model without exhaustive labeleing of the documents. However, they require structural changes to the model that increase the computational burden during sampling. In Section 5.2, I will describe a simpler method to incorporate existing supervision to the verb clustering task.

2.3.6 Dirichlet Processes

The Dirichlet Process (DP) is a generalization of the Dirichlet distribution. It can be thought of as a limiting case where the size of α grows to become infinite. To understand this, consider the fixed-size Dirichlet-Multinomial. In Equation (2.10), we observe a strong rich-get-richer effect. The more times an outcome k is observed, the more likely it is to be drawn from the Dirichlet-Multinomial. α_k acts, essentially, as a smoothing parameter, giving some chance to a topic where $C(k) = 0$. As the number of parameters grows, at some point the Dirichlet-Multinomial doesn't tend to actually "use" the extra clusters. The tendency

for each sample to stick to the clusters with high $C(k)$ dominates[4], and those clusters are used over and over again. This is exactly the behavior that leads to sparse, coherent topics, and makes LDA work.

In the Dirichlet Process, rather than choosing to select one of k topics, each parameter chooses among $k+1$ options - the k known topics, or the option to select a new, currently unused topic. The probability for these options again depends on the count of assignments, but is given as

$$P(x=k) \propto \begin{cases} C(k), & \text{if } C(k) > 0 \\ \alpha, & \text{if } k = k_{new}. \end{cases} \tag{2.7}$$

Note that α still represents a fixed chance of exploring new topics, but as the counts for the known topics grow, it is much less common. Replacing the Dirichlet-Multinomial with a Dirichlet Process allows the topic model to sample new topics as it sees fit, but with a strong tendency to use only a small number. Also, infrequent topics may be abandoned, so the number of topics grows and shrinks slowly during sampling.

The Dirichlet process prior has an advantage over the Dirichlet-Multinomial in that the number of topics does not have to be specified in advance, but instead is selected during sampling. Because this eliminates one of the parameters that must otherwise be selected by the user, these models are often called nonparametric models. The hierarchical Dirichlet process (HDP) (Teh et al., 2006) defines a working mechanism for applying the Dirichlet process in the context of LDA, where each document should have a small number of topics that are shared across documents. However, the HDP is computationally expensive, and impractical on large datasets. However, a DPMM is a much simpler structure that is easier to estimate, and several models discussed in this work use DPMMs to learn the number of clusters.

[4]Of course, this depends on the value of α relative to the counts, but α is typically chosen to be small, because this is desired behavior.

2.3.7 Recap of Bayesian Modeling Background

This section provided a basic introduction to probability (Sections 2.3.1 and 2.3.2) and graphical models (Section 2.3.3). Latent Dirichlet allocation (Section 2.3.4) is a graphical model that learns coherent "topics" from an input corpus, that allow for an interpretable, probabilistic representation of semantics. LDA can be extended with supervision (Section 2.3.5) to make the representation more suitable for recognizing important patterns in the data. LDA uses Dirichlet distributions, and the the quality of the topics depends on the size of distributions used. In many cases, a Dirichlet process can be used instead, to learn an appropriate size while sampling (Section 2.3.6).

2.4 Clustering

Clustering is the task of grouping like items together (Murphy, 2012). It is a fundamental task in machine learning, and has been widely studied. Large datasets of complex objects are hard to reason about, but clustering reduces the number of objects to consider, and the similarities within the clusters can provide insight and useful features for later tasks.

There are many general clustering algorithms that can operate on any dataset with constant, well-defined pairwise item similarities. If the practitioner can define similarity across items in a general, context-free way, any of these algorithms can be easily applied.

The simplest clustering techniques work greedily, starting in a degenerate case and improving it bit by bit. Agglomerative clustering greedily adds links between similar items, starting with each point in a separate cluster. Divisive clustering greedily splits clusters to minimize the similarity between the resulting clusters, starting with all points in the same cluster. The major difference between these greedy clustering approaches is how the notion of pairwise item similarity translates to pairwise cluster similarity. Should the distance between clusters be counted as the minimum pairwise distance across the span, or the maximum, or the average? These produce different clusterings for the same dataset.

k-means, and its corrolary kernel k-means (which depends only on pairwise distances), assume there are a fixed number of clusters (k, a hyperparameter) that each are fairly well-defined and well-separated spheres in some feature space. Each point is assigned to the cluster whose center is nearest, and then the centers are recomputed. If the assumption holds, the clusters can be useful.

Clustering can also be interpreted as a matrix factorization task, operating on the matrix of pairwise similarities (Kuang et al., 2012). Since two items in the same cluster are perfectly similar, according to the clustering output (and two items in different clusters are perfectly dissimilar), every possible clustering produces a matrix of binary pairwise similarities. Clustering can be viewed as reducing the error between the true and the binarized similarity matrices, and there are several approaches that are common in the literature. Symmetric nonnegative matrix factorization (SymNMF) (Kuang et al., 2012) approximates the original distance matrix with positive-only vectors, for each item, and then casts these vectors into binary vectors with exactly one nonzero element to read cluster assignments. Graph factorization clustering (Yu et al., 2006) adds a size-based regularization to this approach, penalizing overly-large clusters. Spectral clustering (Ng et al., 2002) learns the optimal decomposition of the similarity matrix using the eigenvector decomposition, and then applies k-means to the eigenvector embedding of the data points, and so can be viewed as kernel k-means with a particular choice of kernel.

A major drawback to the above clustering algorithms is the notion of a context-free pairwise similarity between all items, which may be required in memory. Probabilistic models like LDA are also clustering algorithms, because each word in the corpus is assigned a latent topic, but that topic depends on the context of the word. Because the topic distributions and document distributions are shared across individual words, and are sufficient for assigning a cluster, the model never needs to explicitly compute similarity between items. To reach an equivalent level of expressiveness, kernel k-means would need to compute a kernel whose dimension was the number of words observed in the corpus. The probabilistic mixture

model lets us compactly specify the context, and how the context affects the cluster, which dramatically reduces the computational burden.

2.4.1 Sense Induction as Clustering

Sense induction is the discovery of the many potential meanings of a word. Hypothesis 2 specifically highlights this issue, and evidence suggests an important facet of verb clustering (Korhonen et al., 2003).

Sense induction can be viewed as a clustering problem. If we take all the examples of a particular word in a corpus, we can assume that they share some small number of meanings. Many words will have only one meaning, but it is easy to find examples of polysemous words. However, if we assign each example to its correct sense, we end up clustering the examples together. In fact, any clustering of the examples corresponds to a possible sense inventory, with one sense per cluster. So, discovering the senses of a word is the same problem as clustering the examples of a word from a corpus. Throughout this work we'll refer to a particular example of a word as an instance.

LDA naturally clusters instances, since each instance of each word is assigned to a unique topic during sampling. Words like "point" may occur in topics of sports terms, or topics of financial terms. Though these topics may be drawn together by "point," the model tries to make the topics concise, so there is pressure from the Dirichlet-Multinomial scheme that drives the topics apart. When coupled with the document-specific distribution's tendency to also be sparse, an instance of "point" with many sports terms co-occurring in the context will effectively ignore the financial topics. This is sense induction, as it learns that "point" means different things depending upon the surrounding context in the document, and assigns those instances to different topics.

In Chapter 3, we will use mixture models with Dirichlet-Multinomial topics to cluster observed instances of verbs into senses.

2.4.2 Verb Clustering Efforts

Following Levin's original paper introducing verb clusters based on syntactic alternations (Levin, 1993), there has been a good deal of work on clustering and classification using these features.

Levin's classes were based on diathesis alternations, essentially the grammatically valid ways to restructure the arguments in a sentence. An example alternation is "Leave a note for her," which has the same meaning as "Leave her a note," but with a different grammatical realizations. Some research has gone into identifying valid diathesis alternations (Lapata, 1999; McCarthy, 2000), although for clustering, identification of diathesis alternations can be approximated (Sun et al., 2013).

Much of the work on verb clustering and diathesis alternation identification relies on the notion of a subcategorization frame (SCF). An SCF is a simplified representation of a sentence, abstracting the grammar from a parser. A SCF from a constituency parser might show the above alternation as "V NP1 for NP2" // "V NP1 NP2". These are related to the syntactic frames used in this work, except we use dependency information rather than constituency information, and we prefer the alternate term to avoid confusion. SCF's have been widely used as features for verb clustering (Lapata and Brew, 1999; Im Walde, 2000; Schulte im Walde and Brew, 2002; Brew and Schulte im Walde, 2002; Sun et al., 2008). Enriching the SCF grammar with selectional preferences, which describe the sorts of nouns likely to take place in the noun phrases, can improve performance (Im Walde, 2006; Sun and Korhonen, 2009).

There has also been interest in classifying verbs into a known, pre-defined set of verb classes. Although these works show the efficiency of syntactic features, most also investigate hand-built linguistic features (Merlo and Stevenson, 2001; Joanis, 2002; Joanis and Stevenson, 2003; Joanis et al., 2008; Lapata and Brew, 2004). Some work investigating the use of these same features suggests that similar features work well for verb clustering, if some supervision is given (Stevenson and Joanis, 2003). The variance across which features per-

formed the best is explained by an investigation by Li and Brew (2008), which concluded that both syntactic and lexical information are useful, but neither performs well alone.

VerbNet is a hierarchical resource, but many of the efforts clustering verbs, including the work in this thesis, flatten the hierarchical structure. One exception notably attempts to recover the hierarchical clustering using hierarchical graph factorization (Sun and Korhonen, 2011). They measure against VerbNet but remove all but the predominant sense of each verb, rather than explicitly tackling the issue of polysemy. Their hierarchical clustering algorithm first makes a large number of fine-grained clusters, and then clusters these fine-grained clusters into larger, high-level clusters. This produces a natural hierarchy of clusters, moving from coarse- to fine-grained distinctions. Any clustering algorithm is amenable to this approach, so if in future work the recovery of the full VerbNet hierarchy is required, this work provides a reasonable starting point.

Some clustering approaches explicitly tackle verb polysemy by allowing the same verb in multiple clusters. A paper by Korhonen et al. (2003) provides a particularly important starting point for the work in this thesis, providing supporting evidence for the hypothesis that polysemy is an important behavior for semantic clustering, and providing a polysemous test set against which some of the work in this thesis is evaluated. This paper uses the information bottleneck (IB) method to cluster directly with the subcategorization frame probabilities. The IB method is based on a criteria from information theory, trying to maximize the mutual information between subcategorization frames and clusters while minimizing the mutual information between the verbs and the clusters. The joining of these two objectives produces clusters that describe informative trends in the data, while penalizing the dominance of clusters which contain only a single verb. This work defines an evaluation set of 110 verb clusters from VerbNet (Kipper et al., 2006), and pulls subcategorization frame assignments from the British National Corpus (BNC), which also provides WordNet senses for each verb. The WordNet senses were manually assigned to gold-standard clusters. The analysis in this paper showed that the automatically-generated clusters were much less ac-

curate when evaluated against monosemous clusters (where each verb is assigned only to the cluster of its predominant sense), which suggests that polysemy in the data prevents alignment to a monosemous gold standard. Further, the authors show that the actual structure of the polysemy is important for semantic clustering, by demonstrating that the quality of alignment with a polysemous training set is significantly better than if the training set were split into senses randomly.

In general, the approaches that handle polysemy the best are probabilistic in nature, and it is beneficial to highlight some relevant probabilistic clustering frameworks. Several papers (Parisien et al., 2008; Parisien and Stevenson, 2010, 2011) use Bayesian methods, including the hierarchical Dirichlet process, to derive verb clusters that share syntactic realizations. These clusters are derived from a corpus of child-directed speech, and are able to predict whether a verb will participate in unseen SCFs after only a small number of observations. This predictive power has already proven helpful to VerbNet annotators including so-called coercive constructions in VerbNet classes (Bonial et al., 2011), which were identified as an important gap in VerbNet's coverage.

Another probabilistic framework that generates semantic verb clusters is called LDA-Frames (Materna, 2012, 2013). The model generates semantic frames, each with a topic distribution over each of its arguments, and is more comparable to FrameNet than VerbNet. However, each frame implicitly defines a cluster of verbs. The model generates semantic frames, each with a topic distribution over each of its arguments, and is more comparable to FrameNet than VerbNet. Each verb participates in a small number of frames, and each frame has a set of slots. Those slots are characterized by a preferences over grammatical relations and semantic roles, which together identify a topic from which to draw the word. This model separates the generation of frames into many independent parts, which is computationally intensive.

An alternative method for generating semantic verb frames using probabilistic models uses a simple Dirichlet process mixture per verb (Kawahara et al., 2014a), but requires a

second verb clustering step to create polysemous verb clusters (Kawahara et al., 2014b). This step-wise method was superior to LDA-Frames at creating polysemous VerbNet-like clusters, and is covered in greater detail in Chapter 3.

Extending VerbNet to other languages is an aspiration of this research, and there has been some interest in creating multi-language verb clusters. While manual work indicates that patterns of polysemy and usage vary across languages (Im Walde, 2006; Majewska et al., 2018), translating English VerbNet to other languages using multi-lingual word embeddings is promising (Vulić et al., 2017).

There have been other efforts to make semantic categories or clusterings of verbs, that have not been focused on Levin's clusters or VerbNet. VerbKB (Wijaya, 2016) pairs verbs with semantic categories from NELL, allowing cross-linking of VerbNet with the large knowledge base and giving additional semantic power to practitioners. Verb Pattern (Cui et al., 2016) introduces a method for discovering finer-grained associations of verbs to semantic roles, and describes an alternative motivation and approach for the verb sense induction task.

2.5 Semantic Vector Models

Computers are optimized for representing vectors of numbers, so there have been a number of approaches attempting to encode semantic information into these vectors. A vector is an ordered list of numbers, essentially coordinates in a general space. This general space is often a representation of co-occurrence matrices, described below in Section 2.5.1. One of the first useful methods for creating semantic vectors is called Latent Semantic Analysis (LSA), which is discussed in Section 2.5.2. This model is based on a matrix of word-document co-occurrences, cleverly scaled. A recent alternative model called Word2Vec (Section 2.5.3) has proven extremely useful at building semantic word vectors, and has been used widely in the last few years to gain improved performance on semantic tasks. Word2Vec uses a different

co-occurrence matrix, a different scaling, and a clever means of compression; these details are described in Section 2.5.4.

2.5.1 Co-occurrence Matrices

Many semantic vectors are based on matrices expressing word occurrences in distinct documents. A corpus with a large number of documents actually gives a very clear picture of the way distinct tokens interact. Each document contains a large number of words, but since a document is written about a single subject, the words that occur together have some sort of semantic relationship: they are both used when discussing that subject. Even ignoring word order (which we know is critical to understanding any sentence), the semantic relationship of simple co-occurence is extremely useful.

Co-occurrence can be expressed as a two-dimensional matrix, where each row corresponds to a single document, and each column corresponds to a single token. The number in the row for document i and token j is simply the number of times token j appeared in document i. Since the documents are generally about a single subject, this matrix is extremely sparse - most of the entries are zero. A few tokens, like "the", "and", and other common functional words, appear in almost every document. But there are many words that are extremely subject-specific, and appear in very few documents (e.g., "zebra", "dovetail", "mousse").

The matrix of document-token counts is a useful starting point for many tasks that are based on capturing meaning, especially after they are compressed. Even though the count of terms in a document does not account for word order, author, or intended audience, it manages to capture a lot of information about each document and reflect important semantic relationships. These matrices are extremely useful for tasks like search (finding documents relevant to a given query), topic discovery, and document classification. Computers can easily represent and manipulate matrices, but the raw co-occurrence matrix is extremely large (often, there are hundreds of thousands of documents and possibly millions of tokens), and extremely sparse. Small, dense matrices are easier to work with and computationally

more efficient, so most approaches start by finding a way to compress this matrix while preserving the same meaning.

2.5.2 Latent Semantic Analysis

Latent semantic analysis (LSA) (Deerwester et al., 1990) is one of the first widely-used techniques for compressing the co-occurrence matrix. It uses the co-occurrence matrix, but scales terms based on document frequency, and compresses the large, sparse representation using matrix factorization. Frequency scaling boosts the importance of terms that are highly predictive of specific subjects. Matrix factorization provides a compact and efficient representation of documents, that still preserves the data.

The document co-occurrence matrix, as described in the previous section, gives very little weight to infrequent terms. A document of two thousand words may have a small number of subject-specific tokens that occur only a handful times, while common function words occur by the score. These common function words, then, account for a large amount of the total count in the row, even though they do little to convey the meaning of the document. Even if we remove the most common words from our matrix entirely, we often see the most predictive terms occuring with low frequency. LSA addresses this issue by multiplying each co-occurrence count ("token frequency" (TF)) with a quantity that boosts words that occur in only a small number of documents ("inverse document frequency" (IDF)). Specifically, the IDF for a word occurring in K out of D documents is computed as

$$\text{IDF} = \log(D/K). \tag{2.8}$$

The IDF reaches its minimum value (zero) when a token appears in every single document. Such a token is not useful for predicting semantics at all, and it reaches its maximum value (which depends on the number of documents in the corpus) when a term appears in one document only. Such a term is extremely likely to be predictive of the subject of that

document. The product of these two terms, or TF-IDF score, gives higher weight to terms that appear often in a document, but also are subject-specific and therefore appear in a small number of documents. TF-IDF is an extremely competitive baseline for semantic tasks.

Compression of large matrices can be accomplished by matrix factorization. If a matrix A can be written as a product of smaller matrices, then the small matrices must convey structural information about A. Singular value decomposition (SVD) expresses A as a product of three matrices,

$$A_{D\times V} = U_{D\times k}\Sigma_{k\times k}V^T_{V\times k}, \tag{2.9}$$

where $k << V$, $UU^T = I$, Σ is diagonal, and $VV^T = I$. This is an extremely well-known technique in linear algebra, and generalizes the eigenvector decomposition to non-square matrices. The matrix Σ contains the largest k singular values, which are essentially the amount of structural information from A that is captured by the corresponding rows of U. The top k singular vectors approximate A more closely than any other set of vectors of the same size, so SVD is optimal from the viewpoint of numerical accuracy.

LSA is simply the application of SVD to the TF-IDF matrix. The vectors in U have only k dimensions (typically k is chosen to be in the low hundreds), and perform on par with or better than the full, uncompressed rows of the TF-IDF matrix for semantic tasks on documents. The columns of V capture important relationships between terms, but are real-valued and hard to interpret, and the rows of V are not very good at representing the meaning of tokens. LSA is not widely used for word-based semantic tasks; it has been most effective in measuring the similarities of document summaries to documents (Landauer et al., 1998).

LDA, covered in Section 2.3.4, can also be viewed as operating on the co-occurrence matrix. Instead of trying to numerically recreate the TF-IDF matrix, LDA attempts to maximize the probability of generating the co-occurrence matrix, subject to the Dirichlet priors on topic and document distributions. A historical bridge between these two algorithms is called probabilistic latent semantic analysis (PLSA) (Hofmann, 1999), which learned doc-

ument and topic distributions to generate the co-occurrence matrix, but did not include Dirichlet priors. The sparseness constraints imposed by the Dirichlet priors significantly improve the coherence of the recovered topics, so LDA has become much more popular than PLSA.

2.5.3 Word2Vec

Word2Vec (Mikolov et al., 2013), and related models like GloVe (Pennington et al., 2014), are extremely good at representing word meaning in a semantic vector. They are inspired by neural language models, because vectors good at predicting the neighboring words must capture this contextual information. Compared to LSA, models in this class of word vectors treat the token, not the document, as the basic unit in the corpus. Each word vector is optimized for how well it predicts the context words that co-occurred within a small window, across the corpus. These vectors carry a lot of semantic information about a word.

Most of what is called machine learning can be cast as an optimization problem. The learning step is to minimize error, or to maximize performance, on the observed data. The function that computes system error is called the objective function. If the objective function can be computed efficiently, quality can be evaluated for any particular solution on the data. If one can also compute the gradient of the objective function (how quickly the solution improves in performance, as the parameters are varied), gradient ascent can quickly find a maximum.

Word2Vec's Skip-Gram model with negative sampling maximizes the following objective function. For each observed token w in the corpus, and each context token c seen within a small window nearby, we add a component to our objective function. We assign a word vector $\mathbf{w}$ for word w, and call the collection of all these vectors $\mathbf{W}$. We similarly define $\mathbf{c}$ and $\mathbf{C}$ for context vectors. Then, our objective to maximize is,

$$L(\mathbf{W}, \mathbf{C}) = \sum_{(w,c)} \sigma(\mathbf{w} \cdot \mathbf{c}) - k\mathbb{E}_c(\sigma(\mathbf{w} \cdot \mathbf{c})). \tag{2.10}$$

σ is the sigmoid function, $\sigma(x) = 1/(1 + e^{-x})$, and $\mathbb{E}$ is the expected value function (which is explained in more detail in Section 2.4). Breaking the above equation into parts, it says the following:

For an observed word, context pair (w, c), we want the vector $\mathbf{w}$ to have a high overlap with the observed context vector $\mathbf{c}$, which we measure using the vector dot product. However, we want the overlap with a randomly selected word to be low, so that we predict the observed context selectively. By subtracting the expected value of $\sigma(\mathbf{w} \cdot \mathbf{c})$ for random c, we penalize $\mathbf{W}$, $\mathbf{C}$ assignments that give high overlap to random words.

Word analogy tasks (e.g. "good" is to "better" as "bad" is to "worse") are used to evaluate the quality of word-based semantic vectors. Word2Vec vectors perform well on this task using simple addition and subtraction: the vectors $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, and $\mathbf{w}$ for those words have the property that $\mathbf{y} + \mathbf{z} - \mathbf{x} \approx \mathbf{w}$. This means that semantic concepts have a shape, a typical size and direction, in the vector space.

Pre-trained Word2Vec vectors are often used as input features for tasks like topic modeling (Batmanghelich et al., 2016). Extensions have been created for analysis of networks (Grover and Leskovec, 2016) and gene sequences (Ng, 2017). In short, Word2Vec vectors are extremely useful semantic representations.

2.5.4 Meaning as a Matrix

Levy and Goldberg (2014) show that the Word2Vec objective function in Equation 2.3 is linked to the pointwise mutual information (PMI) matrix. We can build a co-occurrence matrix, as used in LSA, where we compute counts of (word, context) pairs. In this case, each row represents a token, and the "document" is the total set of all context words observed within a small window in that corpus. As we mentioned before, LSA produces useful vectors for documents. Word2Vec is similar, creating a pseudo-document for each token.

The Skip-Gram objective function reaches its maximal value when, for every word w and

context c,

$$\mathbf{w} \cdot \mathbf{c} = \text{PMI}(w, c) - k = log\Big(\frac{P(w, c)}{P(w)P(c)}\Big) - k. \tag{2.11}$$

k in the above equation is the same k as in Equation 2.3. This value is not the TF-IDF used in LSA, but has broadly similar characteristics. It is large when the word and context co-occur often, but smaller if the context co-occurs often throughout the corpus.

For observed (w, c) pairs, this value is easy to compute, but for unseen pairs it is much harder. A practical solution is to set $PMI(w, c) = 0$ for unseen pairs, and for consistency use only the positive PMI (PPMI) for observed counts,

$$\text{PPMI}(w, c) = \max(0, \log\Big(\frac{P(w, c)}{P(w)P(c)}\Big)). \tag{2.12}$$

The PPMI matrix is easy to compute, and like the TF-IDF matrix it is large and sparse. Following the method of LSA, Levy and Goldberg compute the SVD of the PPMI matrix, and show the rows of the PPMI matrix, and the compressed SVD representations, both perform well on the word analogy task.

Word2Vec owes its performance to the definition of context-based pseudo-documents, and the PMI vectors of these documents. Compared with computing the SVD of the full PPMI matrix, it is extremely efficient to train. Casting Word2Vec as a matrix factorization problem is useful because it is much more suggestive of the ways the input problem can be modified.

Many extensions to Word2Vec can be thought of as modifications to the rows and columns of the PPMI matrix. Paragraph2Vec, or Doc2Vec, adds rows to the matrix that correspond to document co-occurrence counts, exactly as used in LSA. Dependency-based word embeddings swap the context words for labeled dependency relations, and use relationships from a dependency parse, rather than a window of nearby words, as the context mapping. Multilingual Word2Vec can be achieved by adding translations of context words as additional observed contexts. Visualizing these modifications as operations on the PMI matrix makes

it easier to understand the information that will be captured by the final vectors.

2.5.5 Drawbacks of Semantic Vectors

Semantic vectors have been used widely to improve systems like parsing, sentiment analysis, machine translation, and summarization. However, they operate on what is essentially a co-occurrence matrix. Recent improvements have basically resulted from improvements in the construction and scaling of this matrix. However, constructing the matrix is still an open problem with outstanding challenges. If each word is to get a vector, then each word is a row. Likewise, if a word has multiple meanings, we want the word to have multiple rows. This issue of polysemy will be tackled in detail later, but it highlights the main drawback of word vectors: in order to improve our representations, we have to improve our definition of co-occurrence, and that isn't always straightforward. Just because semantic vector models are not mixture models does not mean the problem of polysemy has been ignored. There have been several approaches to making multi-sense word embeddings (Chen et al., 2014; Huang et al., 2012; Neelakantan et al., 2015; Trask et al., 2015). Most approach the problem using prior knowledge, such as the WordNet sense inventory, or try to learn the vectors while updating them. More recent approaches incorporate the context around a word as a part of its word vector, removing the notion of discrete senses by treating each occurrence of a word as a separate event (Peters et al., 2018; Devlin et al., 2018). Chapter 4 provides a model that instead learns semantic topics, providing sense induction and semantic vectors for topics without explicitly calculating word vectors.

2.5.6 Recap of Semantic Vector Background

Semantic vector models are often built on co-occurrence matrices (Section 2.5.1). One of the first widely-used models is called LSA, and is described in Section 2.5.2. LSA is most effective at representing document semantics. A more recent model called Word2Vec (Section 2.5.3) is much more effective for word semantics and is widely used. However, Word2Vec is

essentially a compression of a different co-occurrence matrix with different scaling (Section 2.5.4). The major drawback of semantic vectors is that a matrix compression cannot perform sense disambiguation - this information is needed while the matrix is being constructed (Section 2.5.5).

2.6 Comparison to VerbNet

Creating clusters that align to VerbNet is the driving goal of this research, and we need tools to compare the clusters we create to the true clusters in VerbNet.

2.6.1 Direct comparison of clustered instances

Clustering is a well-studied task, and there are standard clustering metrics to evaluate accuracy. Although there are several choices, we adopt the metrics that have been commonly used in prior works on verb clustering: (modified) purity and inverse purity (Korhonen et al., 2003).

Purity is the proportion of "correct" assignments made by a clustering algorithm, and is analogous to precision in a supervised learning setting. Each found cluster is labeled with the most-frequent true class and any elements of other classes are counted as errors. Precisely, the purity for an induced clustering K of n items, given gold standard classes G is

$$PU(K, G) = \frac{1}{n} \sum_i |K_i| \max_j \frac{|K_i \cap G_j|}{|K_i|}.$$

A perfect purity score may be trivially achieved by assigning each item to its own cluster. Indeed, the purity will always increase with the number of singleton clusters, so the purity may give misleading assessments about cluster quality. For this reason, prior work (Korhonen et al., 2003; Kawahara et al., 2014b) defines modified purity (mPU), where all clusters of

size one are treated as errors,

$$mPU(K, G) = \frac{1}{n} \sum_i I(|K_i|)|K_i| \max_j \frac{|K_i \cap G_j|}{|K_i|},$$

where $I(x)$ is a step indicator function such that $I(x) = 1$ iff $x > 1$, and $I(x) = 0$ otherwise.

The inverse purity (iPU) is analogous to recall, and is computed as

$$iPU(K, G) = \frac{1}{n} \sum_i |G_i| \max_j \frac{|K_j \cap G_i|}{|G_i|}.$$

This metric rewards grouping all items of the same true class together. Singleton clusters in the gold standard G do always count toward increased inverse purity, but there is no reason to penalize the structure of G, or for a modified inverse purity.

2.6.2 Indirect comparison by cluster membership

Some of the papers published in this research used indirect comparisons of clusters (Kawahara et al., 2014b; Peterson et al., 2016). Rather than explicitly clustering the instances from SemLink, we use a normalized metric that allows each verb to participate in multiple clusters. We can compare against a set of polysemous verb classes (Korhonen et al., 2003; Kipper et al., 2006), essentially a subset of VerbNet featuring frequent polysemous verbs, and compare whether the two polysemous clusterings match well.

Because the clustering is polysemous, a typical automatically-induced cluster K will contain only a proper subset of the senses for a particular verb. This partial membership means that cluster assignments for a verb are actually vectors of membership. Define $c_{iv} \in [0, 1]$ as the proportion of instances of verb v grouped into cluster K_i. Again normalizing for errors, the normalized modified purity (nmPU), with respect to the gold standard clusters G, is,

$$\text{nmPU} = \frac{1}{N} \sum_{i \text{ s.t. } |K_i|>1} \max_j \delta_{K_i}(K_i \cap G_j), \tag{2.13}$$

where

$$\delta_{K_i}(K_i \cap G_j) = \sum_{v \in K_i \cap G_j} c_{iv}. \tag{2.14}$$

Similarly, the normalized inverse purity (niPU) is

$$\text{niPU} = \frac{1}{N} \sum_j \max_i \delta_{G_j}(K_i \cap G_j). \tag{2.15}$$

These metrics offer a more holistic view of whether polysemous verb classes are generally aligned, and are reported only in Chapter 5. They were introduced in the Kawahara et al. (2014b) paper.

Chapter 3

Dirichlet-Multinomial Models for Sense Induction and Clustering

The baseline model referred to in this work is from Kawahara et al. (2014b). This model is used because it approaches the task of creation of a VerbNet-like semantic resource using Bayesian mixture models. It is related to earlier work of Parisien and Stevenson (2011), but uses significantly larger corpora and more direct comparison methods. At the time of its publication, it was also the state-of-the-art system on this task, beating several alternate methods on multi-sense alignment (Korhonen et al., 2003; Materna, 2012).

In Section 3.1, we'll examine the Dirichlet process mixture model (DPMM) used in the baseline work for sense induction and clustering. This mixture model places a Dirichlet process prior over Dirichlet-multinomial topics.

This baseline work delivers a working scaffold of semantic verb clustering, breaking the problem into two stages. First, as described in Section 3.2, induce verb senses using an independent DPMM for each verb. Second, as described in Section 3.3, group senses into verb clusters using a second Dirichlet process mixture. These two steps are computed separately, instead of using the hierarchical Dirichlet process (HDP) (Teh et al., 2006; Parisien and Stevenson, 2011), because the two steps benefit from different feature granularity, and

because sampling in the HDP adds significant computational burden compared to treating the tasks as independent. In the preliminary sense induction stage, lexicalized syntactic features (`slot:token` pairs) provide the best verb senses. However, when clustering known senses into verbs, simple `slot` features (aggregating over tokens sharing the same dependency relation) provide the most accurate clustering of verb senses.

Section 3.4 describes a model that performs sense induction and clustering as a single step (Peterson and Palmer, 2018; Peterson et al., 2019). This process is even more similar to LDA, if each document is formed by the sentences from a single verb. The only change required to the LDA algorithm is the restriction that for each instance of a verb observed in the corpus, all the syntactic arguments are assigned to the same topic. This restriction is easy to implement on top of existing, fast, and distributed implementations of LDA (Zaheer et al., 2016; Liu et al., 2011; Řehůřek and Sojka, 2010). Because sharing topics allows semantic generalization, the joint model learns better senses (Section 3.4.1), and the verb clusters align to VerbNet better than the clusters from the step-wise framework (Section 3.4.2).

3.1 Clustering with Dirichlet Process Mixtures

The DPMM used in the baselien framework (Kawahara et al., 2014a,b) is shown in Figure 3.1. The clusters are drawn from a Dirichlet process with hyperparameter α and base distribution G. As discussed in more detail in the background, the Dirichlet process prior allows the model to sample new topics, or remove small ones, as the sampling progresses. Each cluster is chosen proportionally to the number of elements it already contains, i.e.,

$$P(k|\alpha, C(*)) \propto \begin{cases} C(k), & \text{if } C(k) > 0 \\ \alpha, & \text{if } k = k_{new}, \end{cases} \tag{3.1}$$

where $C(k)$ is the count of clustered items already in cluster k. Grouping using this structure is often called the Chinese restaurant process (CRP) (Ferguson, 1973) and encourages a small,

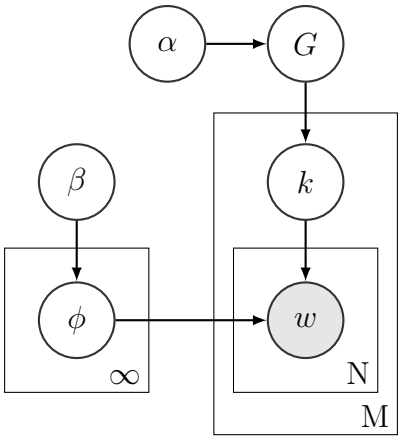

Figure 3.1: The Dirichlet process, Dirichlet-Multinomial mixture model used in (Kawahara et al., 2014a,b) for clustering verb instances and senses. M is the number of verb senses, and N is the sum total of `slot` counts for that verb sense. Each vocabluary item w is drawn by selecting a mixture component k from the Dirichlet Process G, with concentration paramter α. Once k is known, each vocabulary item w is drawn from the corresponding topic ϕ, which has prior β.

but unbounded and unspecified, number of clusters.

Similarly to LDA, the baseline model defines a "topic" for each sense, which is a multinomial distribution over a vocabulary of `slot:token` pairs (e.g., `subj:dog` or `dobj:cat`). Topic distributions are drawn from a Dirichlet with a constant parameter β, which controls the sparseness of the multinomials. When $\beta < 1$, the topics tend to be sparse, which increases the likelihood that the induced senses are coherent, and makes sentences with similar arguments tend to group together. These topics are drawn from the Dirichlet process (DP) prior above.

Each cluster k has an associated multinomial distribution over vocabulary items (e.g. `slot:token` pairs), ϕ_k, which is drawn from G, a Dirichlet distribution of the same size as the vocabulary, parameterized by a constant β. As discussed in LDA, the Dirichlet is a conjugate prior of the multinomial, so we can actually integrate out ϕ_k analytically, given counts of vocabulary items drawn from ϕ_k. For a particular vocabulary item w, we compute

$$P(w|\phi_k, \beta) = \frac{C_{\phi_k}(w) + \beta}{C_{\phi_k}(*) + |V|\beta}, \tag{3.2}$$

where $C_{\phi_k}(w)$ is the number of times w has been drawn from ϕ_k, $C(k) = \sum_i C(*)$, and $|V|$ is the size of the vocabulary.

Since an instance I has many vocabulary items $w_1, w_2, \ldots$, the probability of drawing I is a product of the component probabilities.

$$P(I|\phi_k, \beta) \propto \prod_{w_i \in I} P(w_i|\phi_k, \beta). \tag{3.3}$$

Using Bayes' Theorem, the probability of choosing a cluster k is given by

$$P(k|\alpha, C(*), \phi_k, \beta) \propto P(k|\alpha, C(*)) \prod_w P(w|\phi_k, \beta). \tag{3.4}$$

$\beta < 1$ encourages the clusters to have a sparse representation in the vocabulary space. $\alpha = 1$ is a typical choice, and encourages a small number of clusters to be used.

3.2 Step-wise Sense Induction

The first stage in the step-wise verb clustering is the induction of verb senses (Kawahara et al., 2014a). In this step, the corpus is treated as a set of "instances". Each instance is extracted from a sentence in the corpus, and is a verb together with its labeled dependencies. For the sentence, "The dog chased the cat around the house," the extracted instance would look like `(verb:chase, subj:dog, dobj:cat, prep_around:house)`. Sense induction is treated as a clustering problem, forming groups of instances that share a sense. Polysemy is a per-token event, so the sense induction step is to first partition all instances with the same verb into distinct documents. The clusters are the distinct senses, and the instances are the atoms that are clustered.

Given a large corpus and many examples of a particular verb, inducing verb senses can be accomplished using this Gibbs sampler, as in Algorithm 3.

The baseline framework (Kawahara et al., 2014b) also includes some refinements that

Algorithm 3 Sampling verb senses in the Dirichlet-Multinomial mixture

for verb v in corpus C **do**
  Assign instances to random clusters and compute counts matrices
  for iteration in range(num_iters) **do**
    for instance I with current assignment k in v **do**
      Update counts matrices $C(k)$ and C_{ϕ_k} to remove instance I
      Sample topic k_{new} according to Equation (3.4)
      Update counts matrices $C(k_{new})$ and $C_{\phi_{k_{new}}}$ to add instance I
    end for
  end for
end for

speed convergence. The first is a reduction in vocabulary size by relabeling named entities (e.g., "John", "Microsoft") with a generic <name> token, and clausal complements (e.g., "John thought that <*some other sentence*>") with a generic <CCOMP> token. These replacements have the effect of clustering a subset of the observed arguments on shared semantic properties, and call to mind the improvements from clustering-based selectional preferences in prior verb clustering systems (Sun and Korhonen, 2009). The DPMM will perform its own clustering of tokens, but it may benefit from a more thorough investigation of argument clustering as a preprocessing step. We will come back to this idea in Chapter 6, and highlight it as a direction for future research.

The second is the introduction of "initial frames," which are groups of instances that share the same tokens in the same syntactic slots. First, any sentences that have the same direct object are grouped into a single initial frame. Next, any remaining sentences that share the same subject are grouped into a single initial frame. This process is repeated through a list of syntactic slots, ordered using some linguistic knowledge of the relative saliency of each slot. Rather than clustering each instance using the DPMM, we need only cluster the initial frames. Initial frames with fewer than 10 instances are discarded, so this reduces the number of elements being sampled by more than an order of magnitude. Clustering a smaller number of larger initial frames may speed the DPMM, because the model can mix faster by making larger changes. However, the initial frames remain fixed throughout sampling, which means the model cannot recover from any mistakes in that initialization. This clustering

of sentences is distinct from the supervision efforts in Chapter 5, where we will explicitly cluster sentences with the same known VerbNet labels. But it does have a similar capacity to provide the model with constraints. We will again leave a thorough investigation of clustering sentences as a preprocessing step to future work.

Finally, clustering each verb independently is both a benefit and a drawback. It is much more distributable, since there is no need to synchronize across verbs, and it allows the topics of syntactic arguments to be tailored to the specific usage of each verb. However, not all verbs are frequent, and less-frequent verbs will not necessarily have enough instances to produce coherent topics. We will address this issue in Section 3.4.

3.3 Step-wise Verb Clustering

By separating the verb sense induction and the clustering of verb senses, the features can be optimized for the distinct tasks. According to Kawahara et al. (2014b), the best features for inducing verb classes are joint `slot:token` pairs. For the verb clustering task, `slot` features which ignore the lexical items were the most effective. This supports Hypothesis 1, since the syntactic patterns alone provide the highest-quality clustering.

The second stage treats each induced verb sense as a unit in the clustering, and uses only aggregated syntactic counts. Otherwise the sampling is identical to the sense induction step.

3.4 Joint Sense Induction and Clustering

3.4.1 Sharing Syntactic Contexts Improves Sense Induction

Sharing syntactic topics across verbs yields a considerable improvement in sense induction quality. Treat the collection of instances for a verb as a document, and share the same set of topics, and the sampling is identical to LDA. Actually, it is more proper to use LDA with

a very small modification. An instance may have multiple syntactic slots, but it should be sampled as a unit. For example, an instance (`subj:he`, `dobj:it`, `prep_to:me`) will have to draw all three syntactic slots from the same topic, independently, as in the following equation.

$$P(x_I = k|\theta_k, \beta) \propto \prod_{s_i \in I} P(s_i|\theta_k, \beta), \tag{3.5}$$

where x_I is the topic for instance I, and s_i are the `slot:token` arguments in that instance. The final probability is given by the product of the standard Dirichlet-Multinomial prior for the verb, and this likelihood, i.e.

$$P(x_I = k|I, \alpha, C(*), \theta_k, \beta) \propto P(k|\alpha, C(*)) \cdot P(x_I = k|\theta_k, \beta). \tag{3.6}$$

This is detailed in Algorithm 4.

Algorithm 4 Sampling verb senses with common topics

```
Assign instances to random clusters and compute counts matrices
for iteration in range(num_iters) do
    for verb v in corpus do
        for instance I, with current assignment k, in v do
            Update counts matrices to remove instance I from topic k
            Sample topic k_new according to Equation (3)
            Update counts matrices to add instance I to topic k_new
        end for
    end for
end for
```

This approach is reminiscent of LDA-Frames (Materna, 2012), but is much simpler. Sampling each instance as a unit encourages the topics to represent the entirety of their constituent units, and the `slot:token` vocabulary eliminates the need to sample unique topics for each syntactic slot. This has a huge advantage in ease of implementation because distributed and optimized samplers for LDA are freely available (Liu et al., 2011; Řehůřek and Sojka, 2010). It is similar to the LDA-based selectional preference model of (Wu and Palmer, 2015), which uses LDA on the bag of labeled-dependencies for each verb, and then

uses the resulting topics for Semantic Role Labeling. The contribution here, compared to that work, is to analyze the topics in the context of sense induction; this requirement is the reason to insist that each instance should be assigned to exactly one cluster, even though the prior work did not. The Hierarchical Dirichlet Process (Teh et al., 2006) could also be used here, allowing us to leave the number of topics unspecified, but the algorithm is less practical on large datasets.

Sense induction was run on two datasets. The first, in order to permit direct comparison with prior work, was the Gigaword corpus (Parker et al., 2011). The second is the freely-available Google Books syntactic n-grams corpus (Goldberg and Orwant, 2013). This is possibly the largest dependency-parsed corpus in the English language, and the "verbargs" section neatly aggregates the information about predicate-argument structures. Each line of the verbargs section represents a single pattern of verb and linked dependencies. Because the corpus is so large, these patterns also contain frequencies, and patterns occurring fewer than 10 times are not included. The paper associated with the release of the corpus has much more detail, but the verbargs include the direct syntactic dependencies of each verb pattern, which is exactly the feature set used by the baseline and proposed models.

In order to evaluate the quality of the models, this work employs instances from the SemLink corpus (Palmer, 2009), which has VerbNet class annotation. The test set only contains verbs that occur at least 100 times, in order to ensure there are enough instances to compute meaningful metrics. Also, many of these verbs are polysemous, in the sense of having multiple VerbNet classes, so the test set allows us to evaluate sense induction accuracy.

In order to evaluate the clustering quality, all models are used in the predictive mode. Because the models are generative, they can be used to compute probabilities of sense assignments, even for previously unseen instances. Each labeled instance in SemLink is assigned to its maximum-probability cluster, under the learned parameters of the model.

Table 3.1 shows the clustering mPU, iPU, and F1 score (simple harmonic mean of mPU

Corpus	Algorithm	Verb Sense (Micro)			Verb Sense (Macro)		
		mPU	iPU	F1	mPU	iPU	F1
Gigaword	Step-wise	**85.08**	20.44	32.96	**71.92**	38.72	50.34
	Joint-100	81.59	**43.06**	**56.37**	71.35	**54.07**	**61.52**
	Joint-200	80.29	40.62	53.95	68.16	50.07	57.73
Google	Joint-100	**78.84**	**27.67**	**40.97**	**61.40**	**47.85**	**53.79**
	Joint-200	75.21	26.04	38.68	57.01	44.21	49.80

Table 3.1: Sense induction accuracy, on the Gigaword (**Gigaword**) and Google Books syntactic n-gram **Google** corpora. **Joint-100** refers to the modified LDA algorithm run with 100 topics, **Joint-200** uses 200 topics. The baseline is the maximum-likelihood assignment from the published verb-specific models (Kawahara et al., 2014b), but this baseline is only available on the Gigaword corpus. The highest scores achieved by any model, on each corpus, are highlighted.

and iPU) for senses induced from various models (trained on Gigaword or Google Books syntactic n-grams corpora, with 100 and 200 topics). Since there is a large difference in relative verb frequencies, it is useful to compute micro-average and macro-average of mPU and iPU across verbs. The micro-average weights all instances equally, which gives more weight to accuracy on frequent verbs. The macro-average is the mean of the mPU (or iPU), taken on a verb-by-verb basis.

The major takeaway from this table is that the Joint model outperforms the step-wise model on sense induction, even though the Step-wise model produces a slightly higher mPU. The Joint model does a better job joining sentences that should belong to the same cluster, because it shares topics across verbs. This makes some intuitive sense, because topics that must explain the semantics of many varied verbs are less likely to spend the limited number of available topics to make semantic distinctions relevant to only one verb. This intuition is supported by the fact that increasing the number of topics harms the sense induction performance.

3.4.2 Extending the Vocabulary Improves Single-Step Verb Clustering

This section presents results from a paper currently under review (Peterson and Palmer, 2019), which focuses on adding supervision, and therefore uses only a subset of SemLink for evaluation. Each instance in the evaluation set is assigned its maximum a posteriori topic under the model, and compared against the assigned VerbNet class label. For details on the training/test split of Semlink, and the supervision technique, see Section 5.2.2.

When syntactic topics are shared across verbs, they automatically induce a clustering of verbs, so the model can be viewed as a single-step, joint sense induction and verb clustering algorithm.

One advantage of the step-wise approach to sense induction and clustering is the ability to use different features at each stage. By separating the verb sense induction and the clustering of verb senses, the features can be optimized for the distinct tasks. According to (Kawahara et al., 2014b), the best features for inducing verb classes are joint `slot:token` pairs. For the verb clustering task, `slot` features which ignore the lexical items were the most effective. This aligns with Levin's hypothesis of diathesis alternations - the syntactic contexts are sufficient for the clustering, and encouraging the model to focus on this abstraction helps alignment to Levin-style classes.

The single-step verb clusters also benefit from the combination of syntactic slots and `slot:token` pairs. There are two models to incorporate both sets of features, that are nearly identical mathematically. The first is to make the topic assignment of an instance depend on both a topic distribution over slots and an independent topic distribution over `slot:token` pairs. The second is to simply augment the vocabulary, and count each instance as a bag of `slot:token` pairs and a bag of syntactic slots, effectively double-counting each argument. I implemented the latter approach.

The joint model actually produces better clusters than the step-wise framework, if it has access to the full set of features used by the step-wise approach.

Model	mPU	iPU	F1
Step-wise	48.06 ± 5.00	$\mathbf{54.00} \pm 1.31$	50.69 ± 2.60
Joint	53.42 ± 0.91	43.83 ± 0.31	48.13 ± 0.75
Joint + `slot`	$\mathbf{59.08} \pm 0.39$	46.49 ± 1.64	$\mathbf{52.02} \pm 1.09$

Table 3.2: Clustering accuracy on the test portion of SemLink. Step-wise uses `slot:token` features for sense induction and `slot` features for clustering. Joint shares topics across verbs and performs sense induction and clustering at once. In (Peterson and Palmer, 2018), the model uses only `slot:token` features, but the addition of `slot` features improves the clusters from the Joint model.

Table 5.3 shows that the Joint + `slot` model significantly outperforms the Joint model without `slot` features, even though the improvement over the Step-wise framework does not reach a level of statistical significance. Since `slot` features can improve the model so much, there is further support to the notion that pre-clustering the vocabulary may yield significant gains. We also saw this in Section 3.2, and will come back to it in Chapter 6.

3.5 Recap of Dirichlet-Multinomial Models for Verb Clustering

Sections 3.1, 3.2, and 3.3 present a step-wise framework for verb sense induction and clustering (Kawahara et al., 2014a,b). Section 3.4 shows that a joint sense induction and clustering model can outperform the step-wise framework on both sense induction and clustering tasks (Peterson and Palmer, 2018; Peterson et al., 2019).

Chapter 4

An Alternative Probabilistic Model for Verb Clustering

Section 4.1 describes a novel mixture model that efficiently represents each sense with a semantic vector, which is dramatically faster on the verb clustering task. Section 4.2 describes the features and parameter settings that worked the best. Section 4.3 discusses the limitations of this mixture model, and the properties of the verb clustering task that allow it to be successful.

4.1 A Novel Algorithm for Clustering Senses

The second step of the baseline model is not modeled well by a Dirichlet-Multinomial mixture. Each sense is an aggregate of hundreds or thousands of sentences, but should belong only to a single cluster. Generating a large number of observations from a single distribution is possible, if they are treated as independent, but this is slower than compressing those observations into a single semantic vector.

This chapter describes a The semantic vectors for senses are based on positive pointwise mutual information (PPMI), which was shown to be implicitly related to popular semantic vector models (Levy and Goldberg, 2014). Indeed, the PPMI vectors of word context even

perform well on word analogy tasks, so they seem to be extremely useful semantic representations. The algorithm starts by generating a PPMI vector for each verb sense using the (fixed) counts of syntactic patterns. During sampling, the model computes the count matrices of syntactic patterns assigned to each cluster, and from these counts can easily generate a PPMI vector for each cluster, as well.

Then, given cluster PPMI vectors $\Theta_i, i \in [1, \ldots, K]$, and sense vector x_s

$$P(x_s|\Theta_k) \propto \exp\left(\frac{x_s \cdot \Theta_k}{\tau}\right), \tag{4.1}$$

where $\tau > 0$ is a parameter dictating "temperature", and the probability of assigning sense s to cluster k, with cluster sizes $C_k(*)$, is

$$P(k|s, \Theta, C) \propto P(k|C_k(*))P(x_s|\Theta_k). \tag{4.2}$$

Note that computation of the left factor is given by the Dirichlet process prior.

The inclusion of a temperature parameter allows the model to scale between putting weight on the prior or the likelihood, because it affects how sharply peaked the likelihood term becomes. As the temperature decreases, the maximum-likelihood cluster has more likelihood of being selected, but at high temperatures the likelihood function is close to uniform, leaving the prior as the sole influence on cluster choice. The appropriate setting depends on the data and the typical magnitude of the dot products being computed, and is nontrivial to set. We will come back to this issue in Sections 4.2.2 and 4.3.

The count matrices in this model are identical to the counts tracked using a Dirichlet-Multinomial mixture, so the only new overhead is the computation of PPMI vectors from these matrices. This computation is more effort than the simple smoothing and normalization of the Dirichlet-Multinomial, but by batching samples before updates, this operation is infrequent. During testing, quality does not suffer as the batch size increases, so it suffices to update the vectors only once at the end of each sampling iteration.

Algorithm	`slot` features			`pattern` features		
	mPU	iPU	F1	mPU	iPU	F1
Step-Wise	56.95	28.11	37.64	56.95	28.11	37.64
Joint	46.24	**48.06**	47.14	46.24	48.06	47.14
D-M	48.63	47.77	48.20	36.27	**48.18**	41.38
PPMI	**52.47**	47.34	**49.77**	**60.58**	46.60	**52.68**

Table 4.1: Verb clustering accuracy, for both algorithms, on verb senses from the **Gigaword** dataset. **D-M** is the Dirichlet Multinomial model applied to senses from the Joint framework, and **PPMI** is the novel model proposed here. Two baselines are given, one from the **Step-Wise** framework (Kawahara et al., 2014b), and one from the **Joint** framework in Section 3.4, which does not perform a second pass over the dataset. The highest scores achieved by any model are in bold face. Baseline scores are duplicated in both columns for comparison.

Once the cluster vectors are computed, the likelihoods can be computed at once for an entire batch, and for all topics, with a simple matrix computation. Doing this computation as a batch is not exact, but there is precedent for batching MCMC operations to facilitate distribution (Zaheer et al., 2016). There's no need to add smoothing to the PPMI vectors, and the matrices are sparse. In the end, the runtime is dramatically lower than the Dirichlet-Multinomial model, even with the added overhead of computing PPMI.

Pseudo-code for the PPMI-vector clustering algorithm is given in Algorithm 5

Algorithm 5 Clustering with Exponential Mixture of PPMI Vectors

```
Compute X, the PPMI vectors for input matrix of sense-syntax counts
Assign senses to random clusters and compute counts matrices
Compute and normalize Y, the PPMI vectors for assigned clusters
for iteration in range(num_iters) do
    Compute probabilities by Equation (5), using ⟨X · Y⟩
    Assign new topics to senses and compute counts matrices
    Re-compute and normalize Y
end for
```

Table 4.3 reports runtimes for comparison. Runtimes are measured in seconds, processed on the same single machine with roughly equivalent optimization. A few patterns in the table are worth mentioning. First, the accuracy of the clusters induced by the modified LDA is surprisingly high. On the Gigaword corpus, the mLDA clusters outperform the predictions of the baseline, prior state-of-the-art. The only model that dramatically outperforms this

Algorithm	`slot` features			`pattern` features		
	mPU	iPU	F1	mPU	iPU	F1
Joint	35.04	30.22	32.45	35.04	30.22	32.45
D-M	**44.82**	30.15	**36.05**	**45.96**	30.25	**36.49**
PPMI	14.05	**83.71**	24.07	19.50	**57.99**	29.18

Table 4.2: Verb clustering accuracy, for both algorithms, on verb senses from the **Google** Books syntactic n-grams dataset. **D-M** is the Dirichlet Multinomial model, and **PPMI** is the novel model proposed here. For comparison, the **Joint** model, which skips the second-step clustering is included. The highest scores achieved for each feature set are in bold face. Baseline scores are duplicated in both columns.

Dataset	Features	D-M runtime	PPMI runtime
Gigaword-100	slots	7400	160
Gigaword-100	patterns	6600	280
Gigaword-200	slots	5900	270
Gigaword-200	patterns	9400	320
Google-100	slots	6100	110
Google-100	patterns	4000	150
Google-200	slots	7800	110
Google-200	patterns	4900	140

Table 4.3: Verb clustering runtime (in seconds) on automatically induced senses. The dataset names indicate the corpus and the number of topics used in the sense induction step.

one-shot clustering is the PPMI model using pattern features. Second, the PPMI model always performs better with pattern features over slot features on the same data. Third, the PPMI model is more than an order of magnitude faster.

4.2 Implementational Notes and Feature Engineering

It is sufficient to approximate the Dirichlet process using a Dirichlet distribution with fixed capacity C that exceeds the final number of clusters used, and fixing the concentration parameters at α/C (Kurihara et al., 2007; Ishwaran and Zarepour, 2002). As mentioned above, the Dirichlet-Multinomial becomes sharply peaked when calculating probabilities for senses with thousands of separate observations. The geometric mean of probabilities gives much better performance than the raw product, and helps ensure the scale of the Dirichlet

prior is roughly equivalent. This extra calculation does not seem to be necessary when sampling individual instances in the sense induction step.

4.2.1 Verb Clustering with Syntactic Pattern Features

In the step-wise baseline, the second step is to cluster induced senses into a VerbNet-like structure. In the joint LDA-like approach to the sense induction step, the topics form a natural clustering. A verb sense is a collection of instances with the same verb assigned the same topic; a verb cluster may be defined simply as the collection of verb senses assigned to the same topic. Table 4.1 demonstrates that this verb clustering already outperforms the baseline in F1 score. It also demonstrates that there is a benefit to keeping the second step. In the baseline, the two steps had a key difference. After the sense induction step, the induced verb senses were clustered using a mixture model with a modified vocabulary. Because Levin's distributional hypothesis is based primarily on syntax, `slot` features (which have summed across all tokens sharing the same syntactic slot) gave the best alignment to VerbNet classes.

The syntactic clustering model gains further benefit by introducing `pattern` features, such as `subj/dobj/prep_with`. There is a clear connection between `pattern` features and SCFs used in prior verb clustering work (see Section 2.5.2). The `slot` features worked best in the prior work, but the aggregated count of `subj` and `dobj` counts doesn't give a clear estimate of the number of transitive constructions. It is in general intractable to decipher which arguments occurred together in particular instances. There are more `pattern` combinations than `slot` combinations, but the vocabulary size remains in the thousands.

Tables 4.1 and 4.2 compare the effectiveness of both `pattern` and `slot` features on the verb clustering task. They show results from the D-M model and the PPMI model.

On the Gigaword dataset, the PPMI model outperforms the D-M model and the single-step Joint model. The PPMI model benefits from `pattern` features, and achieves a 5% F1 score improvement over the Joint model from Section 3.4, which uses `slot:token` features.

It is worth noting, however, that the results on the Google dataset undercut this result. All models perform worse when trained on the Google dataset, but the PPMI model actually performs worse than the Joint model, and the D-M is the best-performing model. This seems to be due to a problem with the PPMI model balancing the mPU and iPU tradeoff, which will be discussed in Section 4.3.1. Also, in Table 4.1, the D-M model performs better with the original `slot` features, so the helpfulness of `pattern` features is not universal.

4.2.2 Parameter choices in the PPMI mixture

The parameter τ is important to the convergence properties of the PPMI-based mixture model. It is the same form as the well-known softmax, and as the temperature goes from zero toward infinity, the resulting distribution switches from assigning all probability to the maximum of observed dot products, toward a uniform distribution. Essentially, it governs the extent to which small differences in dot product produce large differences in probability. This is a Bayesian model, so as the temperature increases, the model puts more weight on the prior, which is a CRP with a strong "rich-get-richer effect". We found that $\tau \in [0.01, 1]$ produced reasonable results. Lower temperature values caused the model to make dramatic reassignments frequently, converging quickly, but exhibiting occasional, dramatic shifts of the cluster vectors even after many iterations. Essentially, low temperatures cause the model to choose the maximum likelihood class with extremely high probability, taking greedy steps regardless of cluster size. Larger temperature settings made smaller, more stable steps, and used fewer clusters, because the prior discourages exploration away from the large clusters.

The parameter α behaves exactly as it does in any Dirichlet mixture. It may be tuned to fit the dataset, but the same setting works well for for both PPMI and D-M mixtures.

4.3 When not to use the PPMI mixture

It is worth noting that the PPMI mixture isn't a generative model. The sampling and updating procedures are clear, but there isn't an obvious mathematical process that explains the generation of sparse, non-negative vectors for the original topics. One benefit of LDA is that it encourages sparse topics through its generative model. The PPMI mixture model, in comparison, produces sparse topic vectors by virtue of the PPMI operator on the sufficient statistics matrices. A complete generative model would explain the generation of the observed counts of syntactic patterns, but it is desirable that the model should at least explain the generation of vectors for each topic and sense.

4.3.1 Warnings from Verb Clustering Experiments

Tables 4.1 and 4.2 present different stories about the effectiveness of the PPMI mixture. On the Gigaword corpus, it outperforms all other models. But on the Google Books syntactic n-grams, it does not perform well.

The mPU/iPU tradeoff is governed by the number of clusters the model preferred, and depends more on τ than on α. On the Google Books syntactic n-grams corpus, when setting $\tau > 0.1$ the model tended to use only one or two clusters, extremely favoring iPU. As τ was lowered progressively, the model used more clusters, and purity increased, until $\tau \approx 0.01$. Lowering τ further is impractical. It requires a more complex implementation to avoid numerical overflow after the exponential, but also, the signal from the likelihood overwhelms the Dirichlet prior entirely. Functionally, the knob that governs the mPU/iPU tradeoff reached its maximal setting, without reaching a region of acceptable peformance.

The Google Books syntactic n-grams makes poorer automatic sense distinctions, along with poorer verb clusters. This runs counter to the intuition that a larger corpus should prove more accurate for distributional models. We believe that part of the problem is the elimination of low-frequency syntactic patterns. Pruning of low-frequency items may remove

noise, but it also has a dramatic smoothing effect. Functionally, the Google Books syntactic n-grams corpus represents only a fraction of the original corpus, and and only the smoothest portions remain. It seems that, at least for this purpose, some important signal was lost along with the noise.

This pruning removes many of the sentences that contribute to finer-grained description of automatic senses, as well. The PPMI mixture fares worse using the `slot` features, which are broad and also smooth. With smoothed features on a smoothed corpus, there doesn't seem to be enough discriminative power for the PPMI mixture to tell senses apart. The Dirichlet-Multinomial performs better at making these distinctions, or at least is able to reach a more sensible tradeoff between purity and inverse purity, on this dataset. When using the D-M model on this corpus, the `pattern` features help; they actually harmed performance on the Gigaword corpus.

4.3.2 Failure to Mix on Sense Induction

The verb clustering task is the second part of the step-wise pipeline. Dirichlet-multinomial mixtures worked well for both sense induction and sense clustering, so it is natural to apply the PPMI mixture to sense induction step. In keeping with prior results, we switched from `slot` to `slot:token` vocabulary items, which increased the size of the vocabulary by several orders of magnitude and included a large number of rare terms

These rare terms, and their chance assignments during initialization of the Gibbs sampling procedure, dominated the behavior of the model. While Dirichlet-multinomial mixtures converged to a fairly stable solution in only a few dozen iterations, the PPMI topic model continued mixing and making dramatic changes to many of latent assignments (without seeming to improve topic coherence), after several hundred iterations. Looking at the topic assignments for iterations before and after these changes, the key seems to be in changing the assignments of rare terms. The PPMI topic model is driven to merge or split groups over the presence or absence of relatively rare terms, and all sentences with these terms are

extremely likely to stay in that cluster over time.

The speed advantage of the PPMI topic model also disappeared on the sense induction task. It's possible that after enough iterations the PPMI topic model would converge to high-quality clusters. We observed that the sampling chain did not mix in the time we allowed it, which was longer than the Dirichlet-multinomial model took to produce good verb senses. Also, the Dirichlet-multinomial mixtures work much more quickly on the sense induction task because we can skip the expense of transforming the computation to log space and still not encounter underflow errors.

4.3.3 Mathematical Comparison to Dirichlet-Multinomial

In the step-wise model (Kawahara et al., 2014b), for both the sense induction and verb clustering, multiple observations are drawn from the topic independently. During sense induction, the atoms are sentences, and each `slot:token` pair is an independent observation. During verb clustering, the atoms are collections of hundreds or thousands of sentences, and the corresponding `slot` observations must all be drawn from the same topic. If θ_C is the distribution over clusters for the corpus, and $C_{\mathbf{s}}(w_i)$ counts the number of observations of w_i in sense $\mathbf{s}$, then the probability of assigning $\mathbf{s}$ to cluster k is

$$P(k|\{C_{\mathbf{s}}(w_i)\}_{i=1}^{n}, \theta_C, \phi_k) \propto P(k|\theta_C) \prod_{i=1}^{n} P(w_i|\phi_k)^{C_{\mathbf{s}}(w_i)}. \tag{4.3}$$

This equation is very similar to LDA, with a factor for each observation in the atom.

There are practical consequences to increasing the number of independent observations, which are especially apparent in the verb clustering step because the number of observations is so large. First, the product of so many probabilities can create underflow errors when working with floating point numbers. To get around this, these probabilities are transformed to log space, and transformed back to actually sample. This slows the sampling procedure

quite dramatically. Second, the distribution is sharply peaked, because so many terms are multiplied together, which can slow the progress of the MCMC sampler. This can be alleviated by dividing the counts with the total count of observations in $\mathbf{s}$, $C_{\mathbf{s}}(*)$,

$$P(k|\{C_{\mathbf{s}}(w_i)\}_{i=1}^{n}, \theta_C, \phi_k) \propto \\ P(k|\theta_C) \prod_{i=1}^{n} P(w_i|\phi_k)^{\frac{C_{\mathbf{s}}(w_i)}{C_{\mathbf{s}}(*)}}. \tag{4.4}$$

Working in log space allows this division to happen outside the exponent, ensuring numerical accuracy.

The PPMI topic model's sampling equation can be rewritten analogously to the Dirichlet-Multinomial sampling equation. A dot product is a sum of individual terms, and the exponential of this sum is a product of exponentials. Recall that x_s is the PPMI vector for sense $\mathbf{s}$ and Θ_k is the PPMI vector for cluster k. Let $S(\mathbf{s}, k)$ be the set of terms w for which $x_s(w) > 0$ and $\Theta_k(w) > 0$. Since the terms are products of log functions, we can cancel one logarithm, ending with

$$P(k|\mathbf{s},\theta_C, \mathcal{P}(k)) \propto \\ P(k|\theta_C) \prod_{i \in S(\mathbf{s},k)} \Big(\frac{P(w_i|k)}{P(w_i)}\Big)^{\frac{log\Big(\frac{P(w_i|\mathbf{s})}{P(w_i)}\Big)}{\tau}}. \tag{4.5}$$

This is similar enough to Equation 4.4 to permit some comparisons. The prior is identical, and both likelihoods are products of exponential functions. The first difference is that in Equation 4.5, both the base and exponent involve ratios of probabilities. The second is that we replace $C_{\mathbf{s}}(*)$ with τ to scale the exponent in each likelihood term. And the third is that we don't take the product over all tokens in $\mathbf{s}$, but rather only over the tokens in $S(\mathbf{s}, k)$, which is a strict subset, and in fact guarantee that these ratios are always greater than 1.

After rewriting the sampling equation, it is much easier to see why large vocabularies produce erratic clustering behavior. With large vocabularies, the base probabilities for terms

$P(w_i)$ may be extremely small, and the factor for each term is weighted in both the base and exponent to grow with $1/P(w_i)$. This dramatically boosts the influence of infrequent terms, and is in keeping with observations in the literature that pointwise mutual information places an inappropriately large weight on infrequent terms (Bouma, 2009). The D-M sampling equation, on the other hand, tends to be less influenced by rare terms, because all terms have their base topic preferences weighted equally, and the counts in the exponents are often small for rare words.

4.4 Recap of PPMI Mixture Model

The PPMI mixture model is interesting, but does not generalize well outside the verb clustering task. The verb clustering task is unusual, because it uses a small, expressive vocabulary to cluster large groups of observations (verb senses) into coherent clusters. Verb sense induction has a large vocabulary with a heavy-tailed distribution, and clusters small groups of observations (sentences), which reduces the coherence of the clusters found with the PPMI mixture. This result suggests the PPMI mixture model will be of limited applicability outside the verb clustering task. The PPMI mixture model is only useful for a small set of tasks, those where there is a small vocabulary with strong distinctions, and a large group of observations to cluster.

Chapter 5

Including Partial Supervision to Improve VerbNet Alignment

The theme of this chapter is supervision, which is also the focus of Hypothesis 4. Although a supervised system is less general, and harder to transfer to new domains, it can improve alignment to established VerbNet classes. This increased accuracy is required in order to take advantage of VerbNet's semantic annotation, so the loss in portability is an acceptable price to pay for many applications.

This chapter uses supervision from SemLink, which is covered in Section 2.2.4, and is given in the form of labels for specific instances of verbs from sentences in the Wall Street Journal. We know, objectively, which of these sentences should be clustered together in order to recover VerbNet classes. This is different from the pre-clustering of sentences into initial frames performed in Section 3.2. In both cases we introduce extra information to the models, but in this chapter that information is part of the clustering we hope to recover.

Section 5.1 describes a method of incorporating supervision to the verb clustering step of the step-wise framework Peterson et al. (2016). Because labels are given at a level of individual sentences, not at the level of induced senses, this method depends on sampling an independent class assignment variable, and using aggregated verb statistics to influence

that sampling.

Section 5.2 describes a computationally efficient method to incorporate supervision to the joint sense induction and clustering framweork Peterson et al. (2019). This method observes labeled sentences directly, and uses these observations to bias the sampling toward recovering the known classes. It dramatically reduces variance in the resulting clusters and provides the highest-scoring alignment to VerbNet of any model in this work.

5.1 Indirect Partial Supervision for the Stepwise Model

This section describes a method for adding partial supervision to the step-wise framework. All work in this section is carried out on the baseline set of automatically-induced verb senses. The goal is to make clusters that align to VerbNet, so supervision is only added in the "verb clustering" stage of the two-step model. Section 5.1.1 discusses the mathematical model used to add supervision. Section 5.1.2 discusses modeling choices. Sections 5.1.3 and 5.1.4 compare, quantitatively and qualitatively, the difference in output compared to the baseline model.

When incorporating supervision to the stepwise model, there is a layer of abstraction between labeled sentences and the clusters, because the verb clustering step operates on learned senses. SemLink labels individual instances of verbs, and while we can assign a maximum-probability sense to each of these instances, the sense is comprised of many counts from unlabeled sentences as well. An automatically induced sense of *employ* may have 1,000 unlabeled instances and five labeled sentences. If three of the labeled sentences belong to *Use-105* and the other two belong to *Hire-13.5*, there's no single VerbNet label that correctly captures the sense. Even if all five labeled sentences belong to *Use-105*, the 1,000 unlabeled instances may not be a good reflection of that VerbNet class. This makes assigning a single gold-label VerbNet class to any automatically-induced sense risky and error-prone. Rather than using sentences to label a particular sense, we use them as a distant supervision that

influences the clustering of all the senses of *employ*. For each verb, we add a distribution η describing the likelihood a verb will participate in a particular class, using counts from SemLink. This η is shared across all senses of the verb, and during sampling it encourages verbs in the same VerbNet class to share clusters.

5.1.1 Adding Supervision to a Dirichlet Mixture

Adding supervision to the mixture model is fairly straightforward: at each step, sample both a mixture component k and a VerbNet class y. To accomodate this, each cluster (mixture component) is assigned a a unique distribution ρ over VerbNet classes, drawn from a fixed-size Dirichlet prior with parameter γ. This Dirichlet-Multinomial framework allows easy probability calculations, and has a strong sparsity effect which ensures the mixture components prefer to use a small number of VerbNet classes. In particular, the probability is

$$P(y|\rho_k, \gamma) = \frac{C_k(y) + \gamma}{C_k(*) + |S|\gamma}, \tag{5.1}$$

where $|S|$ is the number of classes in the supervision.

The likelihood of choosing a class for a particular verb requires an estimate of that verb's probability of joining a particular VerbNet class. This starts with initializing η from SemLink, as $\eta(y) = \omega * C_v^{SL}(y) + \delta$, for fixed constants ω and δ, and with $C_v^{SL}(y)$ as the count, in SemLink, of times verb v was assigned to VerbNet class y. From this, it is possible to create a verb-specific distribution θ over VerbNet classes, from a Dirichlet with parameters η, so that η acts as pseudo-counts, steering θ to give high weight to VerbNet classes aligned with SemLink for each verb. This is computed using

$$P(y|\theta, \eta) = \frac{C_v(y) + \eta(y)}{C_v(*) + \sum \eta}, \tag{5.2}$$

where $C_v(y)$ is the number of times verb v is assigned to VerbNet class y by the model.

The VerbNet class for a verb sense is sampled from a product of experts (Hinton, 2002),

the θ_v for the verb v, and ρ_k for the assigned cluster k. This encourages alignment between the VerbNet classes observed in SemLink and the VerbNet classes predicted by the clusters, and is computationally straightforward:

$$P(y|\rho_k, \gamma, \theta_v, \eta) \propto P(y|\rho_k, \gamma)P(y|\theta_v, \eta). \tag{5.3}$$

The product of two distinct probability distributions is called a "product of experts."

Sampling a cluster for a verb sense now depends on the VerbNet class y,

$$P(k|y, \alpha, \phi_k, \beta, \rho_k, \gamma, \theta_v, \eta) \propto \Big(P(k|\alpha, C_k(*)) \times P(y|\rho_k, \gamma, \theta_v, \eta) \times \prod_w P(w|\phi_k, \beta)\Big). \tag{5.4}$$

The supervised process is depicted in Figure 5.1. In brief, each verb v has an η_v, a given by counts from SemLink, which serves as a prior for θ_v. For verb sense, the model samples a cluster k and a VerbNet class y, which depends on θ_v and ρ_k, where ρ_k is the distribution over VerbNet classes in cluster k. ρ_k is drawn from a Dirichlet distribution paramaterized by $\gamma < 1$, encouraging each cluster to have a sparse distribution over VerbNet classes. Because y depends on both θ_v and ρ_k, the clusters are encouraged to align with VerbNet classes.

5.1.2 Modeling Choices

When sampling a cluster for a verb sense with a verb in VerbNet, the model samples y from a product of "experts". θ_v is not incorporated as a prior when sampling y, because there are multiple verbs, with distinct distributions $\theta_{v_1}, \theta_{v_2}, \ldots$.

Because the product-of-experts is a discrete probability distribution, it is easy to marginalize out this variable when sampling k, using

$$P(k|\alpha, \phi_k, \beta, \rho_k, \gamma, \theta) \propto \sum_y P(k|y, \alpha, \phi_k, \beta, \rho_k, \gamma, \theta_v, \eta). \tag{5.5}$$

Either way, once a cluster is selected, the sampler should update the ρ_k and θ_v. Table 5.1.3

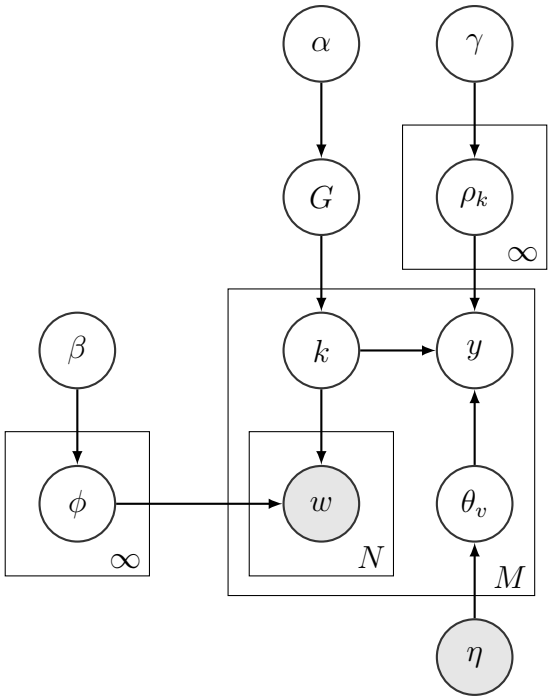

Figure 5.1: The Supervised DPMM used in this work for clustering verb senses. M is the number of verb senses, and N is the sum total of `slot` counts, w, for that verb sense. Each topic ϕ is drawn from a $Dir(\beta)$. Each distribution over VerbNet classes ρ is drawn from $Dir(\gamma)$. G is a Dirichlet process parameterized by α, with a base distribution that combines the vocabulary and supervision priors. θ_v is a verb-specific multinomial distribution over VerbNet classes, and is drawn from a Dirichlet whose parameters η are initialized to reflect the VerbNet class preferences for each verb, when they are known. k is the cluster assignment. A verb like *employ* is known to prefer VerbNet classes *Hire-13.5* and *Use-105*, so each automatically-induced sense of *employ* is more likely to select its variable y from those classes. If a sense has sampled *Use-105*, it will be more likely to select a cluster where the other senses in that cluster (from across all verbs) are likely to have the *Use-105* label.

compares performance for sampling k with assigned y and with marginalized y.

When incorporating supervision, classes are drawn from a flattened VerbNet, using only the top-level categories. This simplifies the selection process for y. In the baseline (Kawahara et al., 2014b), `slot` features were most effective features at producing a VerbNet-like structure; these features are used here as well, for the sake of fair comparison.

Model	nmPU	niPU	F1	N
DPMM	**55.72**	60.33	57.93	522
SDPMM	51.00	**75.71**	**60.95**	122
mSDPMM	51.04	75.00	60.74	129

Table 5.1: Clustering accuracy on verbs in the Korhonen et al. (2003) dataset. N is the number of clusters spanned by the evaluation set.

Model	Example Clusters	
Gold	**push** (0.20), **pull** (0.17)	**give** (1.0), **lend** (1.0), **generate** (0.33), **allow** (0.25), **pull** (0.17), **pour** (0.17)
DPMM	**push** (0.40), **drag** (0.27), **pull** (0.08)	**lend** (0.30), **give** (0.13),
SDPMM	**drag** (0.87), **push** (0.43), **pull** (0.42), **pour** (0.39), **drop** (0.31), **force** (0.09)	**give** (0.82), **pour** (0.02), **ship** (0.002)

Table 5.2: Example clusters from the evaluation dataset (**Gold**), and along with the most-aligned clusters from the unsupervised baseline (**DPMM**) and the semi-supervised clustering scheme (**SDPMM**). Weights given in parentheses describe the total proportion of verb instances assigned to each cluster.

5.1.3 Results

The quality of the supervised model is compared against the clustering from the baseline (Kawahara et al., 2014b). The induced verb senses are used, unmodified, as input, as the supervision step is independent. No supervision is added for verbs in the test set, in order to ensure a fair comparison to the unsupervised model. Table 5.1.3 reports the nPU, niPU, and F1 of the baseline, and the supervised models, against the Korhonen et al. (2003) verb dataset.

Parameters were selected using a grid search, and cross-validation. The results are summarized in Table 5.1.3, comparing the unsupervised DPMM baseline (**DPMM**) to the supervised DPMM (**SDPMM**), and the supervised DPMM sampling k with y marginalized out (**mSDPMM**).

5.1.4 Comparison of Produced Clusters

The supervised sampling scheme produces fewer clusters than the unsupervised baseline. This is in part because it produces fewer "singleton" clusters, containing only one verb sense from the evaluation set. The SDPMM produces only 16% singleton clusters, compared with 34% of singleton clusters from the unsupervised DPMM.

The supervised clusters also tend to cluster more of the senses of each verb into the same cluster. The predominant SDPMM cluster for a verb, which has the highest percentage of a verb's total instances, tends to have 224% the number of instances as the predominant unsupervised DPMM cluster. This tendency does not prevent verbs being assigned multiple clusters. On average, the supervised clustering uses 30% fewer clusters for each verb–a smaller reduction than the 70% overall drop in the number of clusters.

A few example clusters are presented in Table 5.2.

5.2 Direct Partial Supervision for the Joint Model

A more recent paper proposed a single-step, joint sense induction and clustering framework that is nearly identical to LDA, and achieved higher clustering accuracy than the step-wise process. At its most basic, the single-step process is running LDA on a corpus where each document is the collection of sentences with the same verb across the corpus, with dependency labels included in the tokens, and with the constraint that each sentence must be assigned the same topic. This requires only minimal modification to the LDA algorithm, and allows us to use an existing, fast, and distributed implementation for our research. Each topic is a verb cluster, and each topic used by a given verb corresponds to a distinct sense of that verb. An additional benefit of this framework is that the clusters are generated directly from the sentences, with no intermediate steps to obfuscate the labeling. This permits us to develop a scheme for semi-supervised clustering that builds from labeled sentences to higher-quality clusters.

5.2.1 Semi-Supervised Clustering with Direct Observations

To guide our topic model so that its learned topics closely match VerbNet, we explicitly observe some sentences that have a labeled VerbNet class. If our labels span C classes, we use a minimum of C topics, and assign each VerbNet class c_i to a topic k_i. When initializing our topic model, we normally assign each sentence to a random topic, and update the statistics for the Gibbs sampler, which repeatedly updates these topic assignments until convergence. Now, when we initialize our topic model, we also explictly observe some labeled sentences, and assign each sentence with VerbNet class c_i to topic k_i, and we leave this assignment fixed throughout sampling. All unlabeled data is treated normally, at initialization and during sampling.

This is a simple and straightforward means of guiding the clusters, but differs from prior work, which is described in the background section. Verbs with labeled sentences are biased to participate in the correct VerbNet classes, and the topics are biased to contain the vocabulary items corresponding to those same classes. We tune the weight of that bias by observing each labeled sentence w times, because there are orders of magnitude more unlabeled data than labeled, but once the sufficient statistics are initialized there are no further changes to the sampling algorithm.

Our implementation allows the user to specify partial information about VerbNet classes, to help the model conform to this prior knowledge, without requiring a complete specification. Because it does not require any change to the training objective, it creates negligible computational burden. It uses the labeled examples we have, but allows the model room to discover novel classes and novel verb senses, as required to fit the unlabeled data. These are strong advantages over existing work, and our experiments demonstrate it is surprisingly effective.

5.2.2 Evaluation Set

Semlink (Bonial et al., 2013c) provides labels of VerbNet class for each sentence in the Penn Treebank's Wall Street Journal corpus (Marcus et al., 1994). These labels were used to evaluate the quality of sense induction and clustering in prior work, but they are also a potentially valuable resource to guide sense induction.

To test whether a small number of labels can improve the senses learned from LDA, we split this annotation into a training portion and a test portion. The split was designed to address two separate concerns. First, can partial supervision of a VerbNet class improve the recovery of that class from the topic model? Second, can supervision of some known classes aid the recovery of other classes? To address both these concerns, we first split the data by VerbNet class, using 2/3 of the classes as training (hereafter, C_1 denotes the set of classes in the training portion of the split) and 1/3 for testing (C_2). We then split by verb, keeping 2/3 for training and 1/3 for testing. We only use examples from the 141 most-frequent verbs in Semlink, whose labeled sentences span 148 VerbNet classes. This training/test split produced 6400 sentences with known labels for training and 6500 for testing. To permit replicability, we report the training and testing classes and verbs in the appendix.

The primary source of data is Gigaword (Parker et al., 2011), and the Wall Street Journal sections of the Penn Treebank (Marcus et al., 1994), both licensed through the Linguistic Data Consortium. Gigaword is tokenized and dependency parsed automatically as a preprocessing step. Each "document" in LDA is the set of syntactic dependencies observed for a particular verb. The "words" in the document are either syntactic slot labels (*slot*, e.g., "subject", "direct object"), or the concatenation of the syntactic slot and the lexical item observed (*slot:token*, e.g., "subject is John", "direct object is river"). The best model, empirically, uses both sets of vocabulary together, effectively counting each token twice (once with, and once without, the corresponding lexical item). We only consider direct dependencies of the verbs, and prepositional objects labeled with the observed preposition.

When training in the supervised setting, we include the 6400 sentences with known

Model	mPU	iPU	F1
Step-wise	48.06 ± 5.00	54.00 ± 1.31	50.69 ± 2.60
Step-wise + SS	$\mathbf{67.12} \pm 1.88$	54.70 ± 1.17	60.26 ± 1.02
Joint	53.42 ± 0.91	43.83 ± 0.31	48.13 ± 0.75
Joint + `slot`	59.08 ± 0.39	46.49 ± 1.64	52.02 ± 1.09
Joint + SS	64.04 ± 0.38	54.03 ± 0.58	58.61 ± 0.45
Joint + `slot` + SS	65.73 ± 0.49	$\mathbf{62.29} \pm 0.74$	$\mathbf{63.96} \pm 0.58$

Table 5.3: Clustering accuracy on the complete test set, for various models. The Step-wise model with partial supervision (+SS) was the prior state-of-the art for recovering VerbNet classes. The unsupervised Joint model is competitive with Step-wise baseline, especially with the addition of `slot` features. Adding semi-supervision to the Joint model is computationally simpler and ultimately produces a superior result.

labels, and assign each label to a particular topic. These assignments are never re-sampled, so throughout sampling, the supervised verb has some higher-than-random probability mass assigned to the designated topics, and the topics always have some higher-than-random probability mass assigned to the associated vocabulary items. Because Gigaword is much larger than our supervision set, we include a hyperparameter to increase the weight of these labeled instances. Effectively, we label the known sentences as though we'd seen them all many times.

5.2.3 Quantitative Evaluation Protocol

Once a model is trained, we assign each test sentence to its maximum a posteriori topic, and treat all sentences assigned the same topic as belonging to the same cluster. Each test sentence has a correct label, so we have a ground-truth clustering from this annotation. Following the conventions in the literature, we report standard clustering metrics.

5.2.4 Results

The Step-wise model splits sense induction and clustering into independent steps (Kawahara et al., 2014b), and performs on-par with the Joint model which learns senses and clusters simultaneously (Peterson and Palmer, 2018). The Step-wise model uses both `slot:token`

Model	**iPU** on C_1	**iPU** on C_2
Step-wise	52.21 ± 1.90	55.13 ± 1.29
Step-wise + SS	52.88 ± 1.62	55.84 ± 1.14
Joint	47.12 ± 2.56	41.76 ± 1.73
Joint + `slot`	53.48 ± 4.15	42.10 ± 1.94
Joint + SS	58.23 ± 0.99	51.40 ± 0.98
Joint + `slot` + SS	$\mathbf{69.14} \pm 0.27$	$\mathbf{57.99} \pm 1.14$

Table 5.4: Detail of inverse purity for partially-supervised VerbNet classes (C_1), and for never-observed VerbNet classes (C_2), for various models. We expect to recover partially-observed classes better with supervision, but we also see an improvement to recovery of classes that are outside the supervision set.

pairs and `slot` features as vocabulary, and using both sets of features on the Joint model (Joint + `slot`) significantly improves the Joint model results.

Adding partial supervision to these models significantly improves the clustering quality. Supervision in the Step-wise model (Peterson et al., 2016) dramatically boosts the mPU score of the clusters, improving absolute F1 by nearly 10%, and requires a significant increase in computational complexity. Adding supervision to the Joint model using our method significantly improves both mPU and iPU of the clusters, producing a nearly 12% absolute F1 score improvement without increasing computational complexity.

The Joint model with partial supervision, and using both `slot:token` and `slot` features, significantly outperforms all other models at recovering the clustering in our test set.

Adding supervision by biasing particular topics dramatically increases the consistency of the topics learned. In Tables 5.3 and 5.4, we report the mean and standard deviations of the scores across ten runs of each model. Joint + SS models have lower standard deviations, but they are also extremely consistent at recovering nearly the same clusters, run after run, for the seeded topics. Typically in topic modeling applications, different starting conditions produce different clusters, highlighting and obfuscating different themes. However, each seeded topic consistently produced the same clusters. This enhances the practicality of this technique for building VerbNet-style clusters, because adding supervision of new classes should have predictable effects despite the randomized nature of MCMC.

Test Set Verbs	Unsupervised Model		Semi-Supervised Model	
	Frequent Verbs	Test Set Verbs	Frequent Verbs	Test Set Verbs
Calibratable-COS-45.6 rise (536) drop (122) move (56) vary (15) appreciate (4)	increase reduce grow exceed rise	rise (468) drop (49) vary (1) expect (24) drop (8) push (6) grow (6)	*grow* *gain* increase rise reduce *dip* *shift*	rise (512) drop (120) vary (8) drop (29) grow (17) vary (13) hit (5) rise (3) dip (3) count (2)
Use-105 use (588)	use develop support need utilize	use (369) use (5) need (3) call (2)	use need create support have *employ*	use (455) need (46) use (7) call (3) work (2)
Discover-84 find (122)	find find out view work out advise	find (80) find (177)	find *advise* base focus depend	find(122) find (202) work (4) count (3)
Say-37.7 add (282) disclose (155) declare (46) write (26) observe (9)	say tell ask explain add	add (134) declare (9) disclose (2) admit (4) call (2)	*add* convert link subscribe append	add (282) add (3)

Table 5.5: Best clusters from the unsupervised and partially-supervised clustering algorithms for 4 target VerbNet classes. The most-frequent verbs in each cluster are shown, with all terms that seeded the given cluster in the semi-supervised model indicated in italics. We also show the test set verbs assigned to that cluster, with the number of sentences indicated in parentheses. Terms highlighted in red are the model's errors, and show sentences assigned to the cluster that are not in the target VerbNet class. Verbs in both black and red in the same cluster indicate multiple senses of the verb which should have separated into distinct clusters.

5.2.5 Comparison of Produced Clusters

Including the supervision aids recovery of the VerbNet classes in purity and inverse purity. The topic for Calibratable Change of State, from above, was seeded with examples from VerbNet 45.6, including sentences with verbs like *gain, grow, dip* and *shift.* Out of

1042 sentences in the test set, we put 882 of them into the same topic. In the unsupervised setting, we grouped only 594 of these sentences together. We see a similar improvement for VerbNet 105, the Use class. Once seeded with examples from the verb *employ,* we now group 226 of the 362 sentences together, rather than 179.

However, the seeded topics didn't improve everything. We seeded one topic with examples from the VerbNet class Discover-84, using sentences with the verbs *discover* and *hear*. In the evaluation set, the only test examples came from the verb *find*, which is polysemous in the test classes (with examples from Discover-84 and from Get-13.5.1). After supervision, the instances of *find* were clustered together more strongly, and we increased the inverse purity of both Discover-84 and Get-13.5.1. However, in both cases these two classes were incorrectly conflated. Despite a close analysis of the SemLink *find* sentences, it is difficult to account for the placement of *find* in a topic seeded with *advise* sentences.

We also increased the inverse purity score for VerbNet class Say-37.7, which is one of the test classes omitted from the supervision. However, this increase is a result of incorrect lumping with seed examples. The verb *add* is polysemous, belonging to both Say-37.7 ("Elaine added a few words") and Mix-22.1 ("Herman added the computer to the network"), and we included Mix-22.1 examples in our supervision. This produces a cluster that is dominated by *add*, clustered with verbs like *convert* and *link*, which have much lower frequency. Because all examples of *add* end up in this cluster, more examples of Say-37.7 are clustered together after supervision, but we lump them in with Mix-22.1, resulting in a cluster that does not represent the Say-37.7 class. The unsupervised cluster with the most Say-37.7 examples has frequent verbs *say*, *tell*, *ask*, and *explain*, which clearly recovers the desired concept. A similar cluster is created in the partially-supervised clusters, but the test examples from *add* are not included in it.

5.3 Supervision Takeaways

Both models in this chapter leverage a small amount of labeled data alongside a large amount of unlabeled data, and the labeled data has a large positive impact on the learned clusters. This supports Hypothesis 4. The Joint sense induction and clustering model performs best, and receives the most direct and predictable effect of supervised sentences, which supports Hypothesis 3. There is still a large distance to cover with supervision, because we are far from perfect in alignment with VerbNet.

The ability to use a small amount of labeled data to improve the clusters has direct, practical consequences for practitioners, and suggests a fruitful line of research and development of VerbNets in multiple languages. Right now, the task of adding verbs to VerbNet requires reasoning about each verb and each class, and looking through corpus examples to see whether membership is justified (Bonial et al., 2013a). We could instead present annotators with candidate clusters and new candidate members of existing clusters, and allow them to interactively accept or reject those suggestions, and using their decisions to further refine the model's suggestions (Hu et al., 2014). We know that these models respond well to supervision, and we have hints about some potential pitfalls of direct supervision. The seeding can apparently discourage the model from learning to separate senses, or create clusters that are inexplicable in the verbs they lump together, and dealing with these issues will likely present new research problems. The development of effective interactive tools, and best practices for annotating VerbNet, is one of the most promising future lines of research, and will be discussed further in Chapter 6.

5.4 Recap of Supervised Verb Clustering

Supervision can be added to the verb clustering step of the step-wise framework using only information about a verb's typical VerbNet class distribution (Section 5.1). Section 5.1.2 describes a few implementational details. Sections 5.1.3 and 5.1.4 show the quantitative and

qualitative differences between the supervised and unsupervised mixtures. Supervision does improve clustering accuracy, even when that supervision is distant from the VerbNet class variable it ultimately tries to predict.

Supervision can also be added to the joint sense induction and clustering framework (Section 5.2), by directly observing the labels of some annotated senses. This is computationally efficient, makes the recovered classes much more predictable, and produces quantitatively superior results.

Chapter 6

Conclusions and Future Directions

6.1 Conclusions

This thesis presents work on building semantic, VerbNet-like clusters using syntactic features and probabilistic models. We will briefly restate the original contributions from this line of research, the major hypotheses underlying this research, and the evidence supporting those hypotheses.

6.1.1 Review of Contributions

Chapter 3 concerns the use of Dirichlet-Multinomial mixtures for verb sense induction and clustering. Section 3.2 describes the verb-specific sense induction of Kawahara et al. (2014a), and Section 3.3 shows how this lays the groundwork for the step-wise verb clustering system of Kawahara et al. (2014b). A simplified, joint verb sense induction and clustering system (Peterson and Palmer, 2018) is introduced in Section 3.4. This joint sense induction benefits from feature engineering, including raw syntactic `slot` features in the vocabulary Peterson et al. (2019).

Chapter 4 describes a novel positive pointwise mutual information (PPMI) mixture model. This model is effective on the verb clustering task from the step-wise framework

(Peterson and Palmer, 2018), because induced verb senses aggregate syntactic information from hundreds of sentences. Further investigation demonstrated that it does not generalize to sense induction, in part because the PPMI is sensitive to large vocabularies (Peterson and Palmer, 2019).

Chapter 5 describes two approaches to incorporating a limited number of VerbNet annotations to guide the clustering process. Adding supervision on the step-wise verb clustering task uses distant supervision, through sampling an additional variable for VerbNet class assignment, as described in Peterson et al. (2016). In the joint sense induction and clustering framework, a simpler technique is to lock VerbNet classes to particular, pre-assigned clusters, and use the supervision to bias the recovery of these classes. This technique produces superior cluster results, a result which is currently under review as a long paper at ACL 2019 (Peterson et al., 2019).

6.1.2 Review of Hypotheses

Hypothesis 1 *Observable syntactic behavior is a reflection of a verb's semantics.*

This is Levin's hypothesis (Levin, 1993), and underlies all the work in this hypothesis. It has withstood scrutiny in linguistics and by natural language processing researchers, especially practitioners trying to build semantic verb clusters. The apparent coherence of the clusters derived in this research adds further evidence to support this hypothesis.

Hypothesis 2 *Sense disambiguation is a crucial component of deriving semantic groups of verbs.*

Many verbs properly belong in multiple VerbNet classes, because they have multiple senses. Incorrect separation of senses can reduce the semantic coherence of clusters, as can be seen in Table 5.3, where incorrect joining of distinct senses reduces the semantic coherence of clusters in the semi-supervised model.

Hypothesis 3 *Sense disambiguation can be done simultaneously with semantic cluster creation.*

The step-wise sense induction and clustering steps of Kawahara et al. (2014b) require processing the corpus in multiple passes, adding overhead to the clustering task. The joint sense induction and clustering model of Peterson and Palmer (2018) requires only one pass, and produces superior senses and verb clusters. It is also amenable to an extremely efficient supervision scheme (Peterson et al., 2019), so in the presence of labeled data its advantages are even more apparent.

Hypothesis 4 *Partial supervision can increase both the accuracy and the coherence of automatically-created semantic verb clusters, even for clusters with no supervised examples.*

Chapter 5 describes methods of adding supervision to both the step-wise and joint verb clustering models, and evaluates them on a test set specifically designed to test the recovery of classes without any supervision. Although the supervision does not specifically help the step-wise model, it increases the inverse purity (average completeness of recovery) for both seens and unseen classes in the joint model. Table 5.3 specifically highlights model mistakes in coherence, because there is still a large amount of room for improvement. The evidence supports the accuracy claim in this hypothesis, but there is still insufficient evidence that supervision can increase cluster coherence. It may be that coherence improvements require a more complete set of supervised examples. The research in this thesis suggests that annotators can work in tandem with probabilistic models to produce Levin-style clusters that match VerbNet. It is still an open problem how much this human-computer collaboration can improve VerbNet annotation efforts.

6.2 Future Directions

VerbNet has been expanded and revised in the years since the SemLink annotation was done, significantly increasing the coverage of highly frequent verbs and improving the con-

sistency of the classes. A new version of SemLink that reflects these changes is scheduled for release in coming months. We would like to test our system with the new data, as the improved SemLink may produce a further improvement on the clusters.

In Chapters 3 and 4, we noted that some of the models benefit from features with granularity less specific than the particular tokens. The step-wise framework lumps named entities together, without even distinguishing common named entity types (e.g., person, location, organization) (Kawahara et al., 2014b). The joint framework benefits from `slot` features when added as extra counts to the `slot:token` featues it originally used (Peterson et al., 2019). Prior verb clustering efforts have improved from selectional preferences, which were clusters of tokens that gave the model additional information (Sun and Korhonen, 2009). Taken together, these suggest that the features we treat as vocabulary will have a large impact on the model results. Giving the model additional features, or better features, may help it achieve better performance, and in this research we have only scratched the surface of semantically useful representations[1]. We have restricted our model to assign the same cluster to all tokens from the same sentence, and the step-wise framework assigns the same cluster to all tokens from the same initial frame, and then the same cluster to all instances in each verb sense. The atomic unit of the clustering is, then, also a part of the feature representation of the data, and is equally amenable to feature engineering. But there are many plausible feature combinations that are worth trying, and this investigation is left for future research.

Annotators working to build or extend a VerbNet-style resource must make a large number of judgements. The number and natue of senses for each verb, the combination of syntactic alternations that should be viewed as canonical for a verb sense, and whether any particular instance is an idiomatic or otherwise atypical construction are often subtle questions whose answers require care, training, and evidence to answer. The final class assignment that is given by the answers to those questions is equally subtle, and requires the

[1]We did test adding `pattern` features from Chapter 4 to the joint model, but they did not help the way the `slot` features did.

annotator to be intimately familiar with the current classes in VerbNet.

In order to have impact for VerbNet annotation efforts, the model's output must be provided to annotators. Probabilistic models may never be capable of making such fine-grained distinctions, but this work demonstrates they can make reasonable clusters, and that their quality can increase with access to labels. We believe a tool that suggests classes, class members, and provides the annotators with a view into the example sentences would dramatically improve and accelerate their work. The tool could track annotators' decisions, allowing each annotation session to refine and further improve the model's output, interactively (Hu et al., 2014). The most obvious next step for this research is to build that tool, and use it to expand and improve VerbNet in as many languages as possible.

In Section 5.2 we specified supervision at the sentence level, affecting counts for both topics and document distributions by labeling specific sentences, but inference-level supervision of topics can easily be applied to topics or documents without specifying particular sentences. There is use for both broad, topic-level supervision and focused corrections for individual sentences.

Annotators will likely spend time correcting the sense clustering for individual verbs. We saw examples where supervision encouraged distinct senses to be incorrectly linked, and that this created poorer semantic clusters. Because annotation may sometimes cause clusters to lose coherence, or polysemous verbs to have their senses incorrectly lumped, the interactive tool might be able to help users identify errors by viewing specific sentences that have changed cluster between model training sessions. When a batch of annotation decreases coherence, seeing the specific sentences that have changed will allow annotators to correct the trend, either with new labels or by removing problematic annotations. We hypothesize that properly seeding different senses into different classes will dramatically boost VerbNet alignment.

Presenting annotators with an interactive tool for probabilistic verb clustering is only the first step towards a line of research in how to best supervise these probabilistic clustering

models. Interactive guiding of the process can also help us learn more about the nature of the problem. If annotators spend time identifying incorrect lumping or splitting of senses, we may be able to use this signal to quantify the quality of our topics with respect to sense separation. Incorporating such an automatic metric into the statistical model would likely improve it further (Mimno et al., 2011). Part of the benefit to working with annotators will be learning to capture as much value as possible from the signals they give, and this provides an extremely open field of research.

The side effect of research into best practices for VerbNet annotation will be improvements to English VerbNet, and probably a number of VerbNet-style resources in other languages, which will have practical value in addition to the scientific value of the investigation.

Chapter 7

Bibliography

Wikipedia, the free encyclopedia. `http://www.wikipedia.org`.

Aldezabal, I., Aranzabe, M. J., de Ilarraza Sánchez, A. D., and Estarrona, A. (2010). Building the basque propbank. In *LREC*.

Andrzejewski, D., Zhu, X., and Craven, M. (2009). Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *Proceedings of the 26th annual international conference on machine learning*, pages 25–32. ACM.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Batmanghelich, K., Saeedi, A., Narasimhan, K., and Gershman, S. (2016). Nonparametric spherical topic modeling with word embeddings. *arXiv preprint arXiv:1604.00126*.

Bethard, S., Ogren, P., and Becker, L. (2014). Cleartk 2.0: Design patterns for machine learning in uima. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3289–3293, Reykjavik, Iceland. European Language Resources Association (ELRA).

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Bonial, C., Brown, S. W., Hwang, J. D., Parisien, C., Palmer, M., and Stevenson, S. (2011). Incorporating coercive constructions into a verb lexicon. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, pages 72–80. Association for Computational Linguistics.

Bonial, C., Hargraves, O., and Palmer, M. (2013a). Expanding verbnet with sketch engine. In *Proceedings of the 6th International Conference on Generative Approaches to the Lexicon (GL2013)*, pages 44–53.

Bonial, C., Stowe, K., and Palmer, M. (2013b). Renewing and revising semlink. In *GenLex Workshop on Linked Data in Linguistics*.

Bonial, C., Stowe, K., and Palmer, M. (2013c). Renewing and revising semlink. In *Proceedings of the 2nd Workshop on Linked Data in Linguistics (LDL-2013): Representing and linking lexicons, terminologies and other language data*, pages 9–17.

Bouma, G. (2009). Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40.

Brew, C. and Schulte im Walde, S. (2002). Spectral clustering for german verbs. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 117–124. Association for Computational Linguistics.

Chen, D. and Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Chen, X., Liu, Z., and Sun, M. (2014). A unified model for word sense representation and disambiguation. In *EMNLP*, pages 1025–1035.

Christensen, J., Soderland, S., Etzioni, O., et al. (2010). Semantic role labeling for open information extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 52–60. Association for Computational Linguistics.

Cui, W., Zhou, X., Lin, H., Xiao, Y., Wang, H., Hwang, S.-w., and Wang, W. (2016). Verb pattern: A probabilistic semantic representation on verbs. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dowty, D. (1991). Thematic proto-roles and argument selection. *language*, pages 547–619.

Ferguson, T. S. (1973). A bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230.

Giuglea, A.-M. and Moschitti, A. (2006). Semantic role labeling via framenet, verbnet and propbank. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 929–936. Association for Computational Linguistics.

Goldberg, Y. and Orwant, J. (2013). A dataset of syntactic-ngrams over time from a very large corpus of english books.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM.

Hartmann, S., Eckle-Kohler, J., and Gurevych, I. (2016). Generating training data for semantic role labeling based on label transfer from linked lexical resources. *Transactions of the Association for Computational Linguistics*, 4:197–213.

Hautli-Janisz, A., King, T. H., and Ramchand, G. (2015). Encoding event structure in urdu/hindi verbnet. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 25–33.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc.

Hu, Y., Boyd-Graber, J., Satinoff, B., and Smith, A. (2014). Interactive topic modeling. *Machine learning*, 95(3):423–469.

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Im Walde, S. S. (2000). Clustering verbs semantically according to their alternation behaviour. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 747–753. Association for Computational Linguistics.

Im Walde, S. S. (2006). Experiments on the automatic induction of german semantic verb classes. *Computational Linguistics*, 32(2):159–194.

Ishwaran, H. and Zarepour, M. (2002). Exact and approximate sum representations for the dirichlet process. *Canadian Journal of Statistics*, 30(2):269–283.

Jagarlamudi, J., Daumé III, H., and Udupa, R. (2012). Incorporating lexical priors into topic models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 204–213. Association for Computational Linguistics.

Joanis, E. (2002). Automatic verb classification using a general feature space. *Master's thesis, Department of Computer Science, University of Toronto*.

Joanis, E. and Stevenson, S. (2003). A general feature space for automatic verb classification. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 163–170. Association for Computational Linguistics.

Joanis, E., Stevenson, S., and James, D. (2008). A general feature space for automatic verb classification. *Natural Language Engineering*, 14(3):337–367.

Jurafsky, D. and Martin, J. H. (2014). *Speech and language processing*, volume 3. Pearson London.

Kawahara, D., Peterson, D., Popescu, O., and Palmer, M. (2014a). Inducing example-based semantic frames from a massive amount of verb uses. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 58–67.

Kawahara, D., Peterson, D. W., and Palmer, M. (2014b). A step-wise usage-based method for inducing polysemy-aware verb classes. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL2014)*.

Kipper, K., Korhonen, A., Ryant, N., and Palmer, M. (2006). Extending verbnet with novel verb classes. In *LREC*, pages 1027–1032. Citeseer.

Kipper-Schuler, K. (2005). *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. PhD thesis, University of Pennsylvania.

Korhonen, A., Krymolowski, Y., and Marx, Z. (2003). Clustering polysemic subcategorization frame distributions semantically. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL2003)*, pages 64–71.

Kuang, D., Ding, C., and Park, H. (2012). Symmetric nonnegative matrix factorization for graph clustering. In *Proceedings of the 2012 SIAM international conference on data mining*, pages 106–117. SIAM.

Kurihara, K., Welling, M., and Teh, Y. W. (2007). Collapsed variational dirichlet process mixture models. In *IJCAI*, volume 7, pages 2796–2801.

Lacoste-Julien, S., Sha, F., and Jordan, M. I. (2009). Disclda: Discriminative learning for dimensionality reduction and classification. In *Advances in neural information processing systems*, pages 897–904.

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

Lapata, M. (1999). Acquiring lexical generalizations from corpora: A case study for diathesis alternations. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*.

Lapata, M. and Brew, C. (1999). Using subcategorization to resolve verb class ambiguity. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Lapata, M. and Brew, C. (2004). Verb class disambiguation using informative priors. *Computational Linguistics*, 30(1):45–73.

Levin, B. (1993). *English verb classes and alternations: A preliminary investigation*. The University of Chicago Press.

Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Li, J. and Brew, C. (2008). Which are the best features for automatic verb classification. *Proceedings of ACL-08: HLT*, pages 434–442.

Liu, Z., Zhang, Y., Chang, E. Y., and Sun, M. (2011). Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):26.

Majewska, O., McCarthy, D., Vulić, I., and Korhonen, A. (2018). Acquiring verb classes through bottom-up semantic verb clustering. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Manning, C. D., Schütze, H., et al. (1999). *Foundations of statistical natural language processing*, volume 999. MIT Press.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119, Stroudsburg, PA, USA. Association for Computational Linguistics.

Materna, J. (2012). Lda-frames: An unsupervised approach to generating semantic frames. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 376–387. Springer.

Materna, J. (2013). Parameter estimation for lda-frames. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 482–486.

Mcauliffe, J. D. and Blei, D. M. (2008). Supervised topic models. In *Advances in neural information processing systems*, pages 121–128.

McCarthy, D. (2000). Using semantic preferences to identify verbal participation in role switching alternations. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 256–263. Association for Computational Linguistics.

Merlo, P. and Stevenson, S. (2001). Automatic verb classification based on statistical distributions of argument structure. *Computational Linguistics*, 27(3):373–408.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

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

Mimno, D., Wallach, H. M., Talley, E., Leenders, M., and McCallum, A. (2011). Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 262–272. Association for Computational Linguistics.

Moreda, P., Llorens, H., Saquete, E., and Palomar, M. (2011). Combining semantic informa-

tion in question answering systems. *Information Processing & Management*, 47(6):870–885.

Mousser, J. (2010). A large coverage verb taxonomy for arabic. In *LREC*.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Neelakantan, A., Shankar, J., Passos, A., and McCallum, A. (2015). Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856.

Ng, P. (2017). dna2vec: Consistent vector representations of variable-length k-mers. *arXiv preprint arXiv:1701.06279*.

Palmer, M. (2009). Semlink: Linking propbank, verbnet and framenet. In *Proceedings of the generative lexicon conference*, pages 9–15. Pisa Italy.

Palmer, M., Gildea, D., and Xue, N. (2010). Semantic role labeling. *Synthesis Lectures on Human Language Technologies*, 3(1):1–103.

Parisien, C., Fazly, A., and Stevenson, S. (2008). An incremental bayesian model for learning syntactic categories. In *Proceedings of the twelfth conference on computational natural language learning*, pages 89–96. Association for Computational Linguistics.

Parisien, C. and Stevenson, S. (2010). Learning verb alternations in a usage-based bayesian model. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 32.

Parisien, C. and Stevenson, S. (2011). Generalizing between form and meaning using learned verb classes. In *Proceedings of the 33rd Annual Meeting of the Cognitive Science Society (CogSci2011)*.

Parker, R., Graff, D., Kong, J., Chen, K., and Maeda, K. (2011). English gigaword fifth edition ldc2011t07.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Peterson, D. W., Boyd-Graber, J., Palmer, M., and Kawhara, D. (2016). Leveraging verbnet to build corpus-specific verb clusters. ** SEM*, page 102.

Peterson, D. W., Brown, S. W., and Palmer, M. (2019). Verb class induction with partial supervision. In *Review (submitted)*.

Peterson, D. W. and Palmer, M. (2018). Bayesian verb sense clustering. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5398–5405.

Peterson, D. W. and Palmer, M. (2019). Exploring the limitations of the ppmi mixture model. In *Review (submitted)*.

Pradet, Q., Danlos, L., and De Chalendar, G. (2014). Adapting verbnet to french using existing resources. In *LREC'14-Ninth International Conference on Language Resources and Evaluation*.

Ramage, D., Hall, D., Nallapati, R., and Manning, C. D. (2009). Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256. Association for Computational Linguistics.

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

Schulte im Walde, S. and Brew, C. (2002). Inducing german semantic verb classes from purely syntactic subcategorisation information. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 223–230. Association for Computational Linguistics.

Shen, D. and Lapata, M. (2007). Using semantic roles to improve question answering. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*.

Silveira, N., Dozat, T., De Marneffe, M.-C., Bowman, S. R., Connor, M., Bauer, J., and Manning, C. D. (2014). A gold standard dependency corpus for english. In *LREC*, pages 2897–2904.

Stevenson, S. and Joanis, E. (2003). Semi-supervised verb class discovery using noisy features. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 71–78. Association for Computational Linguistics.

Sun, L. and Korhonen, A. (2009). Improving verb clustering with automatically acquired selectional preferences. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 638–647. Association for Computational Linguistics.

Sun, L. and Korhonen, A. (2011). Hierarchical verb clustering using graph factorization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1023–1033. Association for Computational Linguistics.

Sun, L., Korhonen, A., and Krymolowski, Y. (2008). Verb class discovery from rich syntactic data. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 16–27. Springer.

Sun, L., McCarthy, D., and Korhonen, A. (2013). Diathesis alternation approximation for verb clustering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 736–741.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2006). Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476).

Trask, A., Michalak, P., and Liu, J. (2015). sense2vec-a fast and accurate method for word sense disambiguation in neural word embeddings. *arXiv preprint arXiv:1511.06388*.

Vulić, I., Mrkšić, N., and Korhonen, A. (2017). Cross-lingual induction and transfer of verb classes based on word vector space specialisation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2546–2558.

Wijaya, D. T. (2016). *VerbKB: A Knowledge Base of Verbs for Natural Language Understanding*. PhD thesis, Ph. D. Dissertation, Carnegie Mellon University.

Wu, S. and Palmer, M. (2015). Can selectional preferences help automatic semantic role labeling? In ** SEM@ NAACL-HLT*, pages 222–227.

Xie, P., Yang, D., and Xing, E. (2015). Incorporating word correlation knowledge into topic modeling. In *Proceedings of the 2015 conference of the north American chapter of the association for computational linguistics: human language technologies*, pages 725–734.

Yang, Y., Downey, D., and Boyd-Graber, J. (2015). Efficient methods for incorporating knowledge into topic models. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 308–317.

Yu, K., Yu, S., and Tresp, V. (2006). Soft clustering on graphs. In *Advances in neural information processing systems*, pages 1553–1560.

Zaheer, M., Wick, M., Tristan, J.-B., Smola, A., and Steele, G. (2016). Exponential stochastic cellular automata for massively parallel inference. In *Artificial Intelligence and Statistics*, pages 966–975.