The Dissertation Committee for Stephen Creig Roller
certifies that this is the approved version of the following dissertation:

Identifying Lexical Relationships and Entailments with Distributional Semantics

Committee:

Katrin Erk, Supervisor

Raymond Mooney

Risto Miikkulainen

Sebastian Padó

Identifying Lexical Relationships and Entailments with Distributional Semantics

by

Stephen Creig Roller, B.S., M.S.Comp.Sci.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2017

To my sisters, Lindsey and Kacie.

"I don't know the meaning of half those long words, and, what's more, I don't believe you do either!"

– Lewis Carroll, *Alice's Adventures in Wonderland*

Acknowledgments

This research, and this thesis, would not have been possible without the love and sacrifice of the many wonderful people in my life.

Most importantly, I would like to thank Katrin Erk for all the advice given, opportunities afforded, lessons taught, and countless hours spent. She pushed me to try harder and look deeper, and encouraged me through difficult times. I am honored to have worked with her, proud of what we have accomplished, and I am immensely grateful.

I would like to give special thanks to Ray Mooney, Sabine Schulte im Walde, Jason Baldridge, and Gemma Boleda. Each has also acted as a mentor throughout my graduate education, and I believe my perspectives on science, research, and life have all been enhanced by their presence. Thank you also to Risto Miikkulainen and Sebastian Pado, for serving on my committee, as well as their helpful comments and insights.

Additional thanks to all my fellow graduate students for the many discussions and shared experiences, both inside and outside the laboratory. Special thanks to Islam Beltagy, Leif Johnson, Karl Pichotta, Nazneen Rajani, Wesley Tansey, Jesse Thomason, and Subhashini Venugopalan.

Thanks to my parents Bill and Kelly, and my sisters Lindsey and Kacie, for their investment, support, encouragement, and blind belief in my abilities. Finally, thanks to Melanie whose support and sacrifice made this thesis possible, and whose love made this thesis worthwhile.

Stephen Roller
Austin, Texas
April 2017

Identifying Lexical Relationships and Entailments with Distributional Semantics

Publication No. _____

Stephen Creig Roller, Ph.D.

The University of Texas at Austin, 2017

Supervisor: Katrin Erk

Many modern efforts in Natural Language Understanding depend on rich and powerful semantic representations of words. Systems for sophisticated logical and textual reasoning often depend heavily on lexical resources to provide critical information about relationships between words, but these lexical resources are expensive to create and maintain, and are never fully comprehensive. Distributional Semantics has long offered methods for automatically inducing meaning representations from large corpora, with little or no annotation efforts. The resulting representations are valuable proxies of semantic similarity, but simply knowing two words are similar cannot tell us their relationship, or whether one entails the other.

In this thesis, we consider how methods from Distributional Semantics may be applied to the difficult task of lexical entailment, where one must predict whether one word implies another. We approach this by showing contributions in areas of hypernymy detection, lexical relationship prediction, lexical substitution, and textual entailment. We propose novel experimental setups, models, analysis, and interpretations, which ultimate provide us with a better understanding of both the nature of lexical entailment, as well as the information available within distributional representations.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

In modern Natural Language Processing (NLP) research, there is great deal of focus on sophisticated semantic tasks which require complex inference and synthesis of knowledge. These include tasks like Question Answering (QA), where computers must read and answer questions about passages (Hirschman and Gaizauskas, 2001; Allam and Haggag, 2012; Gupta and Gupta, 2012), and Recognizing Textual Entailment (RTE), where computers must decide whether a hypothesis utterance logically follows (or can be inferred) from a given piece of text (Dagan et al., 2006; Marelli et al., 2014; Bowman et al., 2015). In the future, these technologies could influence a wide range of industries: from threat identification in defense, to fact checking in journalism, to synthesis of knowledge in science and medicine.

Substantial progress has been made in systems which perform sentential inferences in QA and RTE, especially as common benchmarks and datasets have become available (Dagan et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009; Marelli et al., 2014; Bowman et al., 2015). Yet in most sophisticated, compositional model of semantics, systems must ultimately consider the semantics of individual lexical items to form a conclusion. This often requires an understanding about the different relationships that can occur between lexical items. Consider the following example:

> Text (Antecedent): The bright girl reads a book.
> Hypothesis (Consequent): A smart child looks at pages of text.

Any language processing system wishing to infer the second sentence from the first must know quite a bit of information about these words: it must know that girl is a kind of child (hypernymy), and that bright and smart have the same meaning in this context (synonymy); that books contain pages of text (meronymy), and that reading involves looking at pages of text (world knowledge).

Although significant progress has been made on the task of Recognizing Textual Entailment, many of these systems ultimately depend on some lexical resources (Beltagy et al., 2014; Bjerva et al., 2014; Lai and Hockenmaier, 2014; Marelli et al., 2014; Beltagy et al., 2016). Possibly the most famous lexical resource is WordNet (Miller, 1995), which organizes the lexicon into a large ontology, though many other resources also exist and are used (Baker et al., 1998; Baroni and Lenci, 2011; Baroni et al., 2012; Ganitkevitch et al., 2013; Jurgens et al., 2012; Levy et al., 2014a; Turney and Mohammad, 2015). Unfortunately, resources as expansive as WordNet are extremely expensive to create, and as language is ever-changing, they are inevitably always incomplete. As such, any dependence on manually constructed resources represents one weak point in some Natural Language Understanding systems. Even recent neural network approaches, which attempt to learn entailments without explicitly depending on these resources, often cannot make entailment predictions about words which were not in the training data (Bowman et al., 2015; Cheng et al., 2016; Pavlick and Callison-Burch, 2016).

Distributional Semantics offers one potential solution to these issues of lexical coverage. Distributional Semantics takes inspiration from the famous quote: "You shall know a word by the company it keeps" (Firth, 1957). In Distributional Semantics, representations of word meaning are automatically induced by counting or modeling the *contexts* in which a word appears. Distributional Semantics is often called Vector Space Models (VSMs) of language, because words are represented as vectors a high-dimensional vector space. Words with similar semantics will have similar vectors in this space. Since VSMs do not require annotated corpora, they are used and studied as an alternative or predictor of particular lexical resources (Baroni et al., 2012; Erk, 2010; Turney and Pantel, 2010).

In this thesis, we consider how VSMs can be leveraged to predict some of the difficult lexical inferences necessary in RTE. Namely, we present techniques and models for predicting specific lexical relationships, entailments, and substitutions using Distributional Semantics. In Lexical Relationship detection, we must predict whether two words exhibit specific, fine-grained linguistic relationships, like hypernymy (is-a relationships; e.g. 'cat' and 'animal') or meronymy (part-whole

relationships; e.g. ‘cat’ and ‘tail’). In Lexical Entailment detection, we must predict a coarser entailment label, such as simply entailing or non-entailing, without fine-grained labels of linguistic relationship. In Lexical Substitution, we must propose a context-specific synonym for a word in a given sentential context, such that must consider a word’s multiple meanings.

We consider each of these tasks, in turn, and present novel models based on distributional approaches of word meaning. Furthermore, we further demonstrate the value of our efforts by integrating our techniques into a larger framework of sentential semantics, and show improvements on a real, end-to-end RTE task, especially compared to a model which uses only a fixed lexical resource.

While each of these tasks is approached and tested for empirical contributions, we also aim to understand the improvements and benefits offered by each of our models across the various tasks. To this end, each empirical contribution is additionally complemented by an analysis of *why* the contribution is attained. In this way, we hope we can show not just how distributional semantics can enable lexical entailment, but how lexical entailment provides insight about distributional representations, and the information contained within them.

1.1 Thesis Outline

The remainder of this thesis is structured as follows:

In Chapter 2, we introduce the necessary background information to understand this thesis, including a brief discussion of Distributional Vector Space models and their construction. We also briefly introduce each of the relevant tasks considered in this thesis, including hypernymy detection, lexical relationship prediction, lexical substitution, and Recognizing Textual Entailment.

In Chapter 3, we consider the importance of experimental setup in hypernymy detection, and propose an experimental setup which better measures how models generalize to novel lexical items. We propose Asym, a Supervised Distributional model for hypernymy detection, and show it outperforms prior work in our experimental setup. Finally, we examine Asym analytically and show it does

not exhibit prototypicality behavior, where a model makes predictions like "cactus is an animal," only because animal is a prototypical hypernym, and cactus is a prototypical hyponym.

In Chapter 4, we dive deeper into a hypernymy detection model which is known to exhibit strong prototypicality behavior. We propose a novel analysis of this model by interpreting the model within the framework of *context vectors*, and show it learns to identify distributional contexts related to Hearst patterns. We propose a novel model based on this behavior, which extends a prototypicality classifier using an iterative procedure similar to Principal Component Analysis (PCA). We evaluate our model in hypernymy detection, general lexical entailment detection, and lexical relationship prediction, and show our model matches or outperforms several existing models in the literature.

In Chapter 5, we propose two novel, unsupervised models for the Lexical Substitution task, where one must propose a synonym of a target word in a given sentential context, which preserves the meaning of the sentence. We show our models quantitatively outperform a similar model across three datasets of Lexical Substitution, especially when models must propose Lexical Substitutes from the entire vocabulary. An analysis of our models show they prefer to predict high-frequency words as substitutes, enabling them to prefer correct substitutes over misspelled alternatives.

In Chapter 6, we integrate the findings of the previous chapters into a real, end-to-end RTE system, showing that our contributions in lexical semantics are useful in a complex task requiring sentence-level reasoning. An error analysis of RTE shows the different components succeed and fail in ways consistent with the findings of other chapters.

We conclude our thesis in Chapter 7, summarizing our contributions and findings, and proposing some directions for further investigation.

1.2 List of Thesis Contributions

In this thesis, we make the following contributions:

Experimental Setup for Predicting Lexical Relationships

We observe the importance of experimental setup in lexical relationship classification and hypernymy detection, noting that traditional train/test splits can lead to *lexical memorization* and overestimation of the lexical generalization. We propose an alternative experimental setup for lexical relationship classification, where there is zero lexical overlap between the training and test sets. For example, if the word 'animal' may not appear in both the training set and test set. We find this experimental setup is considerably more difficult, and better measures the ability of models to generalize to novel lexical items.

The Asym model for Hypernymy Prediction

We propose Asym, a supervised distributional model for hypernymy detection. We connect Asym to the Distributional Inclusion Hypothesis, which has motivated a large number of unsupervised models of hypernymy detection. We show the Asym model performs strongly in zero lexical overlap settings, We analyze Asym within the context of the prototypicality framework proposed in prior work, and find it does not simply learn prototypicality behavior.

H-features for Lexical Relationship Prediction

We analyze a lexical relationship classifier which is known to exhibit strong prototypicality behavior, wherein a model predicts 'cactus $\rightarrow$ animal', simply because 'animal' is a prototypical hypernym. We consider its performance with respect to different distributional spaces, and use a novel technique to interpret the model with respect to distributional contexts. We observe that it predicts relationships using distributional contexts based on well-known Hearst patterns. We argue that this prototypicality behavior is interesting and valuable for the task of relationship prediction.

We propose a novel lexical relationship classifier called H-features, which exploits prototypicality behavior, and extends modeling power using an iterative

PCA-like procedure for feature extraction. This new H-feature model obtains strong or state-of-the-art performance across nine datasets constituting a variety of linguistic relationships. We show that the model remains interpretable across this array of linguistic relationships, and learns to look for interesting distributional contexts beyond Hearst patterns.

The PIC and nPIC models for Lexical Substitution

We propose *nPIC*, a novel unsupervised model for Lexical Substitution. Our model builds off of another simple model of Lexical Substitution, by replacing simple cosine similarity with an unnormalized dot product. Our Lexical Substitution model outperforms a comparable Lexical Substitution mode on three datasets. Our improvements are most pronounced in a difficult evaluation task, where a model must propose substitutes from the entire vocabulary. We also propose *PIC*, which includes additional parameters learned in a self-supervised manner. We find the additional parameters provide significant gains in objective performance on Lexical Substitution across all three datasets.

We analyze our models of Lexical Substitution, and show our model prefers substitutions with higher unigram frequencies, enabling it to disregard misspelled substitutes like 'colorfull (sic)' in favor of their correct counterparts. We also find that the additional parameters of the PIC model further exaggerate this unigram bias.

Integrations into an RTE system

We evaluate lexical entailment classifiers within the framework of a complete end-to-end RTE system, and observe that strong performance in lexical entailment is a critical component for a successful RTE system.

We observe that the probability of a word pair entailing in RTE is non-monotonic in the cosine similarity between the words: the pair is more likely entailing the higher its cosine similarity, until the highest levels of cosine similarity, at which point words are more likely to be co-hyponyms and therefore non-entailing.

We integrate this observation into the RTE system and show improvements at both the lexical level and RTE level.

Finally, we integrate the contributions from our chapters on Asym, H-features, and Lexical Substitution into the end-to-end RTE system. We find that many of the improvements observed in earlier chapters contribute to a higher score in the end-to-end RTE system, providing additional validation for the contributions of this thesis.

Chapter 2
Background

2.1 Chapter Overview

In this chapter, we review some of the background critical to this dissertation. We begin with a discussion of Recognizing Textual Entailment (RTE) as core motivation of our thesis. We then overview Distributional Semantics, outlining its purpose and two specific implementations employed in this thesis. Finally, we discuss the Lexical Entailment (LexEnt) and Lexical Substitution (LexSub) tasks, which we view as two useful proxies for the kinds of lexical semantics necessary in RTE. We do not argue that these tasks are completely sufficient, but one goal of our thesis to show that developments in these tasks improve practical RTE.

2.2 Recognizing Textual Entailment

In the previous chapter, we introduced the Recognizing Textual Entailment (RTE) task as a challenging semantic problem in the field of Natural Language Processing. One of the first benchmark papers describes RTE as "recognizing, given two text fragments, whether the meaning of one text can be inferred (entailed) from the other" (Dagan et al., 2006). For example, an RTE system could read the sentence "A bright girl reads a book," and predict it entails "A smart child looks at text."

Since this original definition, many other datasets (Giampiccolo et al., 2007; Bentivogli et al., 2009; Marelli et al., 2014) and countless approaches have been described (c.f. Dagan et al. (2013) for a thorough survey). Although RTE is not usually considered an end-user application by itself, successful RTE systems may influence many downstream tasks like Information Extraction or Question Answering and become useful in information-heavy industries like defense, journalism, and science.

RTE is a very difficult task, at least partially due to its generality. True entail-

ment reasoning may require a mix of common sense and domain-specific knowledge; sophisticated logical inference about coreference, scope, quantifiers and implications; and a substantial ability to judge the importance and relationships between individual lexical items (Dagan et al., 2006). Modern RTE systems do *not* attempt to cover every possible entailment phenomenon, but do strive to make incremental progress on individual problems with varying approaches. In this thesis, we focus predominantly on some of the issues of *lexical semantics* necessary for reasoning in RTE systems. Specifically, we consider issues like lexical relationship detection, where one must classify *how* two words are (or are not) related, and lexical paraphrasing, where one must suggest alternative words which have the same meaning.

Presently, there exist many approaches of RTE systems. Early systems for RTE focus heavily on detecting word overlap or alignments between the text and hypothesis. These systems may focus on looking for word re-use between the two sentences, or the presence of predictive words in either sentence, like 'not' (Dagan et al., 2006). Modern, deep learning approaches focus on learning word and sentence representations from large datasets (Bowman et al., 2015; Cheng et al., 2016; Bowman et al., 2016; Mou et al., 2016; Rocktäschel et al., 2016; Vendrov et al., 2016). These models attempt to learn all RTE entailments and word relationships implicitly through a large dataset of annotated RTE examples. While they perform incredibly well statistically, recently it has been noted that these systems boil down to sophisticated variations of the word alignment idea (Parikh et al., 2016), and sometimes fail to learn basic properties about lexical meaning (Pavlick and Callison-Burch, 2016), and therefore may fail unpredictably, especially with unseen words.

Other kinds of RTE approaches focus more heavily on doing sorts of logical, deductive reasoning correctly, but depend heavily on large lexical resources to capture the implications and information necessary for lexical reasoning (MacCartney and Manning, 2008; Bjerva et al., 2014; Beltagy et al., 2016). These rich lexical resources, including WordNet (Miller, 1995) and PPDB (Ganitkevitch et al., 2013), provide an excellent source of common sense knowledge and word relation-

ships which can be used as a background knowledge-base during logical reasoning. However, when a word exists outside the knowledge-base, or the relationship between two words is not labeled, then the systems will also fail to draw even simple conclusions; as such these systems may have very high precision, at the extreme cost of recall.

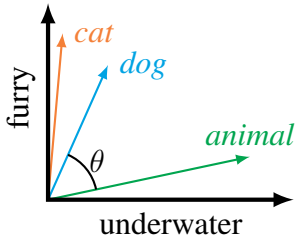
In our work, we consider whether it is possible to improve upon this latter class of RTE systems by distilling information about lexical relationships automatically. We turn now to Distributional Semantics and Vector Space Models, which provide an automatic induction of word meaning using only large, unannotated corpora.

2.3 Distributional Semantics

Distributional Semantics is a powerful tool for automatically inducing semantic representations for lexical items (Turney and Pantel, 2010; Erk, 2012). The core notion is that of the *Distributional Hypothesis*, that if two words appear in similar *contexts*, they can be assumed to have similar meaning. This idea has a long history in the linguistic and philosophical literature that can be traced back over 60 years (Wittgenstein, 1953; Harris, 1954; Firth, 1957). In its modern form, Distributional Semantics involves finding *vector space representations* of words which are constructed by counting or modeling the contexts in which a particular word appears. According to the Distributional Hypothesis, words with similar *vectors* can be assumed to have similar *meanings* (Turney and Pantel, 2010). For this reason, they are often referred to as Vector Space Models (VSMs) of language. Variations on this idea have also become immensely popular in the neural networks community, with algorithms like Skip-gram Negative Sampling (SGNS) (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), and have often replaced traditional count-based VSMs in the NLP community (Baroni et al., 2014). In their neural forms, these distributional vectors are often referred to as *word embeddings*, because they *embed* the word in a fixed-dimensional vector space. In this dissertation, we freely use the terms *embeddings* and *word vectors* interchangeably, and we will

```
         the furry dog is friendly to
and manipulate the dog 's lips and
       as a clever dog ; two to
a reputation among dog trainers of having
    also among the dog breeds most likely
 the very earliest dog shows and kennel
        as a guard dog and to hunt
   the mechanic 's dog began to howl
```

(a) toy corpus



(b) cartoon vector space

Figure 2.1: (a) Example contexts of the word 'dog', and (b) Cartoon drawing of word vectors in the `furry` and `underwater` dimensions.

cover the SGNS algorithm in more detail in Section 2.3.3.

In its simplest form, vectors are induced by defining a vector space where each dimension in the space corresponds to a particular context word. A large, unannotated corpus of text is then processed, finding instances of a target word, like 'dog', and incrementing a count for each of the target's *co-occurrences*, or words appearing around the target word 'dog', as in Figure 2.1. With a large enough corpus, coherent statistical patterns begin to form. For example, the word `furry` is likely to be used to describe both 'cat' and 'dog', which is then reflected in the vector counts (Lund and Burgess, 1996). After constructing vector representations for the words 'cat' and 'dog', we can then compare these vectors using various geometric distance metrics, most prominently *cosine similarity*:

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\sum_i u_i v_i}{\sqrt{\sum_i u_i^2}\sqrt{\sum_i v_i^2}} = \frac{\mathbf{u}^\top \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} \tag{2.1}$$

Here, i iterates over all the different context dimensions, like `furry` or `kennel`, and cosine similarity is defined over the range $[-1, 1]$ or $[0, 1]$, depending on the exact distributional space chosen. Words with similar vectors will have a smaller angle between them, and therefore a higher cosine similarity (i.e. closer to 1). For example, in one of the distributional spaces used in this thesis, 'brother' and 'sister' have a cosine similarity of 0.85, but 'brother' and 'truck' have a cosine similarity of only 0.14.

2.3.1 Count Transformations

In practice, usually the distributional vectors are more sophisticated in their construction than raw co-occurrence counts. Typically, words and contexts below a certain threshold are omitted from the co-occurrence matrix, because extremely rare words have few counts and therefore impoverished representations (Turney and Pantel, 2010). The co-occurrence matrix is also usually transformed using some nonlinearity; one common choice is Positive Pointwise Mutual Information (PPMI) (Bullinaria and Levy, 2007), where the raw co-occurrence count between a word w and context c is transformed,

$$\text{PPMI}(w, c) = \max\left(0, \log \frac{P(w, c)}{P(w)P(c)}\right)$$

Pointwise Mutual Information (PMI) measures roughly how many times more likely two items co-occur more often than chance, while Positive PMI additionally ignores co-occurrences that occur less often than chance. Other transformations, like conditional probability ($P(w|c)$ or $P(c|w)$) (Hofmann, 1999; Blei et al., 2003), Local Mutual Information (LMI) (Evert, 2005), and soft-plus ($\log(1+x)$) (Pennington et al., 2014), are also sometimes seen in the literature, and emphasize different aspects of lexical similarity.

2.3.2 Context Selection, and Syntactic Contexts

Defining contexts is another important aspect of distributional semantics. In the example of Figure 2.1, we showed that context can be defined as three words to the left and right of the target word, but there are alternatives. For example, using very large windows of co-occurrence (or even entire documents) results in emphasizing more *topical* similarity, e.g. doctor and hospital, while smaller windows emphasize more *functional* similarity, e.g. doctor and surgeon (Peirsman, 2008; Agirre et al., 2009).

Context can be also defined as *syntactic neighbors* extracted from a dependency parse. For example, in Figure 2.2, the contexts for the word *chased* would

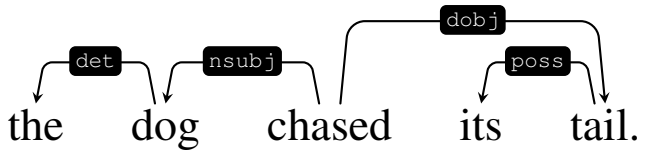


Figure 2.2: Example of a dependency parse for "The dog chased its tail." In a syntactic distributional space, contexts are defined as adjacent nodes with their labeled edges.

be nsubj+dog and dobj+tail. Distributional Spaces defined in this manner tend to emphasize the *selectional preferences* of words, or the tendency of words to have particular arguments in their syntactic relations. (Padó and Lapata, 2007; Baroni and Lenci, 2010; Levy and Goldberg, 2014a). For example, the subject of *barks* is likely to be *dog*, while the subject of *purrs* is likely to be *cat*.

2.3.3 Dimensionality Reduction

Dimensionality Reduction is another important aspect of Distributional Semantics. As described earlier, distributional vector spaces are very high-dimensional: bag-of-words spaces have many thousands of dimensions (Turney and Pantel, 2010; Mikolov et al., 2013; Pennington et al., 2014), while syntactic spaces may have millions (Baroni and Lenci, 2010; Padó and Lapata, 2007). Efficiently dealing with these large, extremely sparse vectors can be troublesome, so we often opt to use some form of *dimensionality reduction* in the form of matrix factorization. In dimensionality reduction, the co-occurrence matrix $\mathbf{M}$ is assumed to be factorizable into two lower-rank matrices,

$$\mathbf{M} \approx \mathbf{V}\mathbf{C}^\top, \tag{2.2}$$

where $\mathbf{V}$ is some lower-rank representation of word vectors, and $\mathbf{C}$ is the corresponding lower-rank representation of the context items. These projections of words and contexts into the same latent space traces back to the earliest days of distributional semantics (Landauer and Dumais, 1997; Deerwester et al., 1990), and is critical to many of the contributions of this thesis.

Singular Value Decomposition

One common form of matrix factorization used in the literature is the Singular Value Decomposition. In this formulation, we find a factorization

$$\mathbf{M} = \mathbf{A}\mathbf{\Sigma}\mathbf{B}^\top,$$

where $\mathbf{A}$ and $\mathbf{B}$ are unitary matrices, and $\mathbf{\Sigma}$ is a diagonal matrix. These three matrices are used to derive the $\mathbf{V}$ and $\mathbf{C}$ matrices,

$$\mathbf{V} = \mathbf{A}\mathbf{\Sigma},$$
$$\mathbf{C} = \mathbf{B}.$$

The SVD is particularly important because it is uniquely defined for any matrix, has a direct connection to the eigendecomposition, and has natural or domain-specific interpretations. One especially useful property is that the diagonal matrix $\mathbf{\Sigma}$ contains an *ordered* list of the *singular values* of the matrix. Thus, in its complete form, the $\mathbf{\Sigma}$ matrix will actually contain as many entries as the rank of the matrix, but we arbitrarily pick the first k singular values in order to find a low-rank approximation (Trefethen and Bau III, 1997).

Skip-Gram Negative Sampling (Word2Vec)

More recently, alternative forms of distributional semantics have made a large impact on the NLP community, in particular the Skip-Gram Negative Sampling (SGNS) model, commonly known as *Word2Vec* (Mikolov et al., 2013). Like traditional count-based distributional semantics, SGNS aims to learn a vector representation for words where words with similar meaning have similar vectors. It came into wide popularity after the famous observation that simple vector arithmetic could be used to do some analogical reasoning: for example, Mikolov et al.

(2013) famously observed that

$$\text{king} - \text{man} + \text{woman} \approx \text{queen}.$$

SGNS is a log-bilinear model of co-occurrence, where the model learns *input embeddings* representing words and *output embeddings* representing contexts. These embeddings are k-dimensional vectors, typically about $k = 300$, as with other dimensionality reduction models. The Word2Vec algorithm learns embeddings in a single pass over a corpus, and does not require storing any large, sparse co-occurrence matrix, making it much faster and memory thrifty than traditional count-based models. The algorithm learns by observing each target word w with a context c, just as Section 2.3.2. For each co-occurrence observed, the model pretends that it must perform a classification task, predicting whether the observed co-occurrence is either real, or random noise according to a log-bilinear model similar to logistic regression,

$$P(w|c) = \sigma(\mathbf{w}^\top \mathbf{c}), \tag{2.3}$$

where σ is the logistic function,

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2.4}$$

To turn this it a real classification problem, for each real observed word w, several *negative samples* are also "observed," by stochastically selecting several words $\bar{w}_1, \ldots, \bar{w}_n$ from the unigram distribution. The model should then predict a high probability for the true observation $P(w|c)$ and low probabilities for $P(\bar{w}_i|c), i = 1, \ldots, n$. In each observation, the parameters $\mathbf{w}$ and $\mathbf{c}$ are updated via Stochastic Gradient Descent (SGD). With a large corpus, many iterations of SGD are run, resulting in high-quality representations for $\mathbf{w}$ and $\mathbf{c}$.

Initially, the community was perplexed as to why representations learned by this algorithm should be so much better than those from traditional count-based methods (Baroni et al., 2014). However, almost all of its mysteries have since been explained: the SGNS model itself is implicitly equivalent to PMI matrix factoriza-

tion (Levy and Goldberg, 2014b); its improvements over count-based models are mostly due to heuristics (Levy et al., 2015a); and its analogical reasoning is related to a balance of cosine similarities (Levy et al., 2014b). Nonetheless, today the model is often the typical "default" choice of NLP practitioners, due to its extreme efficiency and strong efficacy.

2.3.4 Problems with Distributional Representations

The Distributional Hypothesis and its computational implementations have proven to be extremely useful in practical applications of NLP (Cho, 2015; Goldberg, 2016). Indeed, one may even argue that the most important and impressive advancements of the past several years are fundamentally powered through the distributional hypothesis (Manning, 2015). Nonetheless, distributional models do have important limitations.

Recall above we showed that the cosine similarity of 'brother' and 'sister' is 0.85, while 'brother' and 'truck' was only 0.14. However, the words 'hot' and 'cold' have a similarity of 0.89, even more than that of 'brother' and 'sister'. Though the meanings of 'hot' and 'cold' are highly related, they are natural antonyms, and it is (generally) impossible for something to be both hot and cold simultaneously. Thus we can tell that words have related meanings, but not *how* or *why* they are related, or *what* is the relationship between them. Additionally, many measures of distributional similarity, like cosine similarity, are inherently symmetric, or that

$$\cos(\mathbf{u}, \mathbf{v}) = \cos(\mathbf{v}, \mathbf{u}),$$

thus it is impossible to detect any asymmetric relationship using only cosine similarity. In the next section, we explore some of these relationships, and will address these problems in greater detail throughout the thesis.

2.4 Lexical Entailment and Relationship Detection

Zhitomirsky-Geffet and Dagan (2005) define Lexical Entailment as any in-

stance where one word "can be substituted by [another word], such that the meaning of the original word may be inferred from the new one." More generally, Lexical Entailment is any semantic relation where the meaning of one word is implied by the meaning of another (Shnarch, 2008). This includes many classical lexical relations, like hypernymy (e.g. 'girl' is a 'child'; a 'dog' is an 'animal'), and meronymy (e.g. 'girl' has 'eyes'; a 'dog' has a 'tail'), but it can also include a wide variety other inferences which are difficult to categorize, like 'to snore' implies 'to sleep'.

As shown in our Introduction example, understanding and predicting these lexical relationships is critical to performing certain inferences in RTE: without basic lexical relationships, even the easiest textual entailments would be out of reach. There has been a great deal of research around predicting lexical relationships automatically from text. We cannot possibly enumerate all the work on this problem, but we aim to cover some influential approaches and to emphasize attempts related to distributional semantics

One important, early development in this task was Hearst patterns (Hearst, 1992), which are specific textual patterns highly indicative of particular relationships. Common Hearst patterns include exemplar phrases like "animals such as cats," "animals including cats," which are both highly indicative of hypernymy. Later, Snow et al. (2004) extended this Hearst pattern approach to use syntactic patterns. By using syntactic parses, some longer distance patterns are more easily captured, like "animals such as cats and dogs," which implies "animals such as dogs." Girju et al. (2006) proposed a variety Hearst patterns for meronymy (a part-whole relation), including a variety of genitive constructions like "baby's eyes," "eyes of the baby," and "bird without wings."

More recently, groups have begun researching how lexical relationships may be mined automatically using VSMs. Since Distributional Semantics provides a way of estimating word meaning automatically from only large, unannotated corpora, they may also be able to identify word relationships (Baroni and Lenci, 2011; Baroni et al., 2012). Ideally, this could be used to augment existing lexical resources like WordNet, bootstrap a WordNet-like resource in new languages, and help downstream tasks like RTE and QA.

Early work in predicting lexical entailments using distributional spaces was focused mostly on attempts to find unsupervised similarity measures to identify hypernymy from word vectors (Weeds et al., 2004; Clarke, 2009; Kotlerman et al., 2010; Lenci and Benotto, 2012; Santus, 2013). The reasoning was that with the right corpus, the right distributional space, and the right similarity measure, hypernym pairs (or at least candidate pairs) could be readily identified using only word vectors. This view was developed in part by evidence that the ubiquitous cosine similarity tends to highlight co-hyponym pairs more than other relations (Weeds et al., 2004; Baroni and Lenci, 2011). One lasting hypothesis about hypernymy detection has been the Distributional Inclusion Hypothesis (DIH) (Zhitomirsky-Geffet and Dagan, 2005), which states that the contexts in which a hypernym appears should be a superset of all its hyponyms. A considerable amount of work assumed the DIH to be at least partially true, and many of the proposed measures were based on the Distributional Inclusion Hypothesis in one form or another (Clarke, 2009), or a hybrid of DIH and cosine similarity (Kotlerman et al., 2010; Lenci and Benotto, 2012).

As it became obvious that unsupervised measures did not work as well as hoped, the community began working on entailment detection as a supervised task. Baroni et al. (2012) proposed, as a preliminary baseline of a novel dataset, training a simple baseline classifier to predict whether word pairs were either hypernyms or non-hypernyms. Although they reported strong performance, others later realized their model struggled with issues of *lexical memorization*, or a special kind of overfitting (Roller et al., 2014; Weeds et al., 2014; Levy et al., 2015b). As such, more recent works have emphasized their performance when *individual words are held out entirely*, so that the same word can never appear in both training and testing sets (Roller et al., 2014; Kruszewski et al., 2015; Levy et al., 2015b; Shwartz et al., 2016; Roller and Erk, 2016a). We discuss more about this issue of Lexical Memorization in Chapter 3.

2.5 Lexical Substitution and Polysemy

In our discussion of lexical relationships in the previous section, we assumed that words are *monosemous*, or that they have only one meaning. However, words like 'bright' have multiple meanings, which change depending on context, as in 'bright girl' and 'bright coat'. When a word has multiple meanings, it is *polysemous*, and polysemy is one of the fundamental difficulties with understanding natural language (as well as a source of beauty in so many poems and jokes).

Handling polysemy is one of the oldest problems in NLP, and probably one that remains fundamentally unsolved. Lexicographers often attempt to simply catalog and enumerate the different senses of a word, as one sees in the Oxford English Dictionary, or in the different *synsets* in WordNet. This has led to the very important task of Word Sense Disambiguation (WSD), where one must identify which sense of a word is being used in a particular context or sentence (McCarthy, 2009; Navigli, 2009).

However, lexicographers may not always agree on how fine-grained to consider senses: obviously a 'river bank' and a 'financial bank' are very different word senses, but should we distinguish the many different kinds of financial banks, like investment banks, deposit banks, and federal banks? This leads inevitably to the flip-side of the WSD: Word Sense Induction (WSI), where one must take a dataset of a word's unannotated usages, and then induct or identify all its possible sense meanings (McCarthy, 2009; Navigli, 2009). It is clear how that the WSD and WSI may be important and useful task in the search of lexical semantics, but other possibilities exist.

One approach to dealing with polysemy is to model how word meaning *shifts* in a given context; that is, we can explicitly model what happens to a word based on its use in a given context (Erk and Padó, 2008; Erk, 2010). Since 2007, one way to measure this has been the Lexical Substitution task. In the Lexical Substitution task, we are provided with a sentential context, and must suggest *substitutes* which can replace the given target word, while preserving the meaning of the entire sentence (McCarthy and Navigli, 2007; Biemann, 2012; Kremer et al., 2014).

At first glance, the Lexical Substitution task has a less obvious connection to Textual Entailment than the Lexical Entailment task does. However, we argue it is also an important proxy to improvements on Textual Entailment, and that Lexical Substitution may act as a kind of lexical entailment *in context*: if a substitute can replace the target and preserve the meaning of the sentence, then it follows that the target *entails* the substitute. Although this includes basic synonymy (like 'bright' and 'clever'), it also covers much more interesting specialized cases. For example, with the context

> "Tara stood stock-still, waiting for the first tiny gleam from the scout craft to appear in the darkness of the **wormhole**,"

human annotators considered *portal* and *rift* to be excellent substitutes. These substitutes take into account the Science Fiction context of the sentence, indicating the task is more complicated than simple synonymy. Indeed, Kremer et al. (2014) found that only 9% of substitutes in their dataset were direct synonyms in WordNet. For this reason, we believe that Textual Entailment can benefit more from modeling Lexical Substitution as a complete task, rather treating the phenomenon as explicit lexical relations.

Distributional Semantics offers a tempting solution to this problem of Lexical Substitution, given its ability to measure the *graded* levels of similarity between words (Erk and Padó, 2008). Interestingly, although there have been both supervised (Biemann, 2012; Szarvas et al., 2013a) and unsupervised attempts (Erk and Padó, 2008; Dinu and Lapata, 2010; Thater et al., 2010; Van de Cruys et al., 2011; Kremer et al., 2014; Melamud et al., 2015a,b; Kawakami and Dyer, 2016; Roller and Erk, 2016a) using distributional semantics, presently unsupervised measures hold an edge (Melamud et al., 2015a, 2016).

2.6 Chapter Summary

In this chapter, we examined the task of Recognizing Textual Entailment, emphasizing its difficulty and the wide variety of phenomena it encompasses. We

briefly considered two broad kinds of RTE systems and described how they often approach lexical semantics. We then described the Distributional Hypothesis, and its computational realization, Distributional Semantics. Our discussion of Distributional Semantics covered how word vectors are derived from unannotated corpora (using both count-based and prediction-based techniques), and how similarity measures like cosine may be used to words with related meanings, but that they fail to consider how words are related. We then considered a variety of lexical relationships, like hypernymy, synonymy, co-hyponymy, and their importance in RTE. We also considered issues of polysemy, its difficulty, and how it can also be related to the RTE task.

Chapter 3
Hypernymy Detection and Asym

This chapter introduces the Asym model and related material. The work in this chapter has been published in Roller et al. (2014), and all work in this chapter constitutes original contributions.

3.1 Chapter Overview

Automatically identifying lexical relationships is a long standing task in NLP (Hearst, 1992; Snow et al., 2004; Girju et al., 2006), which has implications for textual entailment, question answering, information retrieval, and more. Hypernymy, or an is-a relationship like 'cat' is an animal, is of particular interest for many applications. Most approaches tend to focus on identifying specific lexico-syntactic patterns which are indicative of particular relations, but require that direct observation of words in those patterns. The use of distributional vectors holds promise of increasing recall over lexico-syntactic patterns, but the literature has focused mostly on unsupervised measures with limited success.

We propose Asym, a novel *supervised* model for automatically identifying hypernymy using distributional vectors. Our measure draws inspiration from the Distributional Inclusion Hypothesis (Zhitomirsky-Geffet and Dagan, 2005), which states that the contexts in which 'animal' may appear should be a superset of the contexts in which 'cat' may appear. We propose a novel experimental setup to measure how well models generalize to unseen lexical items, and show that our model performs well in these adversarial experimental conditions.

3.2 Prior Work

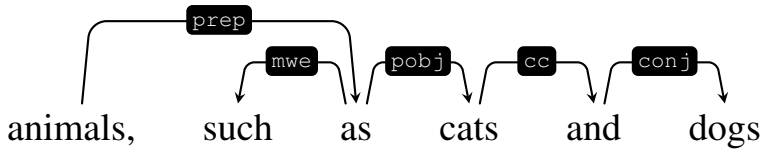
There is a large body of work on automatically extracting word relationships from unannotated corpora, and thus it is nearly impossible to cover all work exhaus-

tively. Nonetheless, we here try to sketch out some of the main branches of research which have led to our own work.

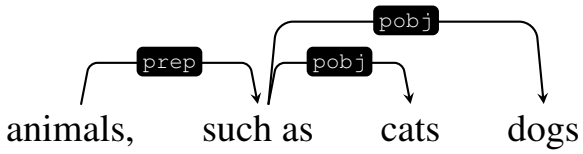
One of the seminal works in automatic hypernymy discovery was that of Hearst patterns (Hearst, 1992). Hearst showed that certain lexico-syntactic idioms were very clear indicators of hypernymy relationships. Some example Hearst patterns are 'animals such as cats'; 'animals for example cats'; and 'animals including cats'. Each of these provides evidence that 'animal' is a hypernym of 'cat', especially when the pattern is repeated several times across a large corpus (Hearst, 1992). With a strong set of patterns, and a large enough corpus, many hypernymy relationships may easily be captured.

Unfortunately, such patterns can be brittle and sensitive to minor inflections or variations. These patterns above would fail to also capture the relationship between 'animals' and 'dogs' if provided 'animals, such as cats and dogs'. Some work has sought to address these issues by expanding the lists of Hearst patterns and increasing coverage by using rough frequency estimates from web search engines (e.g. Google, Bing) rather than explicit counts (Seitner et al., 2016). Others generalize known patterns over Part-of-Speech tags (Tjong Kim Sang, 2007) or language models (Ritter et al., 2009). These efforts can greatly improve on the low recall of Hearst patterns with relatively small sacrifices in precision.

Another line of work seeks to generalize Hearst patterns from word patterns to syntactic patterns. Snow et al. (2004), in particular, showed that using patterns over syntactic parses can greatly improve the representational power, and significantly improve both precision and recall. For example, Figure 3.1 shows how a dependency parse can be lightly processed so that the conjunction 'and' no longer stands between 'animals' and 'dog', and the multiword expression (MWE) 'such as' is collapsed to a single node. Later work by Snow et al. (2006) shows this can be improved further by jointly identifying many relations in a taxonomy at a single time. However, even with improvements through joint relationship identification and more patterns, one limitation of Hearst patterns is that words *must* co-occur together explicitly using one of the patterns. This limitation, combined with the success of distributional vector space models, has led to the question: can distribu-



(a) vanilla dependency parse



(b) collapsed dependency parse

Figure 3.1: (a) Dependency parse for "Animals, such as cats and dogs," and (b) The collapsed dependency parse of the same (de Marneffe and Manning, 2008).

tional methods be used for identifying lexical relationships? And if so, what can their success teach us about distributional methods?

3.2.1 Distributional Approaches

Much of the earlier work in applying distributional approaches to lexical entailment detection focused primarily on *unsupervised* methods, positing that relationships should be apparent through regular behavior across the dimensions of a sparse vector space. The reasoning went that with the right entailment similarity measure, one could measure the entailment-similarity between any two words and estimate the degree to which one word entailed another. This would occur in the same way that cosine similarity can tell you the degree to which two words are related. Crucially, the hypothetical *entails* measure *must* in general be asymmetric,

$$\text{entails}(\mathbf{u}, \mathbf{v}) \neq \text{entails}(\mathbf{v}, \mathbf{u}),$$

or any two entailing terms (e.g. 'cat' $\rightarrow$ 'animal') will also entail the converse ('animal' $\rightarrow$ 'cat').

One of the first approaches was that of Weeds et al. (2004), who introduced the notion of *distributional generality*, where v is distributionally more general than

u if the u appears in a subset of the contexts in which v is found, as illustrated in Figure 3.2. They speculate that hypernyms should be more distributionally general than hyponym, and measure distributional generality using a variation of precision (Weeds and Weir, 2003; Weeds et al., 2004):

$$1(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

$$\text{WeedsPrec}(\mathbf{u}, \mathbf{v}) = \frac{\sum_{i=1}^{n} u_i * 1(v_i)}{\sum_{i=1}^{n} u_i} \tag{3.2}$$

Intuitively, *WeedsPrec* computes the number of contexts of u which are also a part of v, and weights each context by its importance to u. Note that the weights of contexts in v are *not* considered, only their absence or presence.

Geffet and Dagan (2004); Zhitomirsky-Geffet and Dagan (2005) point out that generic distributional similarity is too loose for precision-demanding applications like Question Answering or Textual Entailment, and that the definition should be changed slightly. They propose the *Distributional Inclusion Hypothesis* (DIH), which states that a *lexical entailment* relation holds between two words when "one of the words can be substituted by the other [word], such that the meaning of the original word can be inferred from the new one" (Zhitomirsky-Geffet and Dagan, 2005). The DIH generalizes the view of distributional generality beyond hypernymy and provides a more rigorous definition. Figure 3.2 illustrates the DIH, emphasizing that the contexts in which 'cat' appears should be a subset of the contexts in which 'animal' appears. Meanwhile, under this interpretation, co-hyponyms should have a high degree of overlapping contexts.

To this end, Kotlerman et al. (2010) introduces a dataset for lexical entailment and propose the distributional similarity measure *balAPinc* predicting lexical entailment. The *balAPinc* measure is a modification of the Average Precision (AP) measure from Information Retrieval. The general notion is that scores should increase both with the number of included features, and give more weight to the highly ranked features of the narrower term u. This is captured by computing the

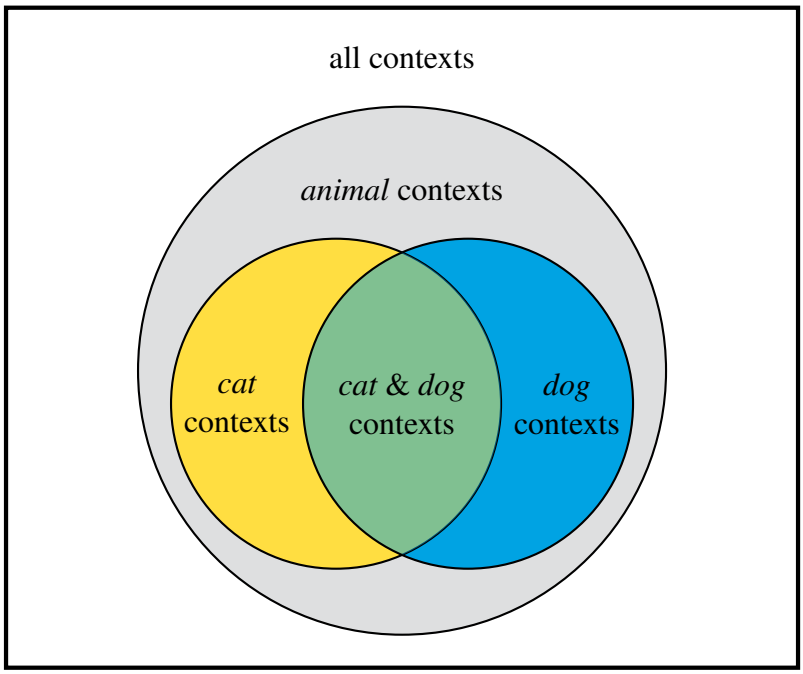


Figure 3.2: Illustration of the Distributional Inclusion Hypothesis (DIH) (Zhitomirsky-Geffet and Dagan, 2005), which hypothesizes that the contexts in which 'cat' appears should be a subset of the contexts in which 'animal' appears. Co-hyponyms are hypothesized to have overlapping, but distinct contexts.

number of included features at every rank $P(r), r \in \{1, \ldots, |W|\}$, and weighting by the corresponding rank in the broader term, $rel'(f_r)$. The final measure, *balAP-inc* smooths using the well known *LIN* similarity measure (Lin, 1998):

$$\text{APinc}(\mathbf{u}, \mathbf{v}) = \frac{\sum_{r=1}^{|1(\mathbf{u})|} P(r) \cdot rel'(f_r))}{|1(\mathbf{u})|} \tag{3.3}$$

$$\text{balAPinc}(\mathbf{u}, \mathbf{v}) = \sqrt{\text{APinc}(\mathbf{u}, \mathbf{v}) \cdot \text{LIN}(\mathbf{u}, \mathbf{v})} \tag{3.4}$$

Another measure proposed in the same vein is *ClarkeDE* (Clarke, 2009), which takes inspiration from Weeds et al. (2004), but also considers the *degree* to which a term v is contained within u by evaluating over a component-wise minimum. Intuitively, this allows us to measure distributional inclusion beyond mere presence or absence (as in *WeedsPrec*), and consider the degree to which *important* contexts are included.

$$\text{CL}(\mathbf{u}, \mathbf{v}) = \frac{\sum_{i=1}^{n} \min(u_i, v_i)}{\sum_{i=1}^{n} u_i} \tag{3.5}$$

Lenci and Benotto (2012) introduce the *invCL* measure which uses *ClarkeDE* to measure both distributional inclusion of u in v and distributional non-inclusion of v in u. While all other measures interpret the Distributional Inclusion Hypothesis as the degree to which a $\subseteq$ relation holds, Lenci and Benotto (2012) test the degree to which proper inclusion $\subsetneq$ holds. They consider not only the degree to which the contexts of the narrower terms are included in the contexts of the wider term, but also determine the degree to which the wider term has contexts that the narrower term does not have, and balance the two terms using a geometric mean:

$$\text{invCL}(\mathbf{u}, \mathbf{v}) = \sqrt{\text{CL}(\mathbf{u}, \mathbf{v}) * (1 - \text{CL}(\mathbf{v}, \mathbf{u}))} \tag{3.6}$$

A few other unsupervised approaches, not based on the DIH, have also been proposed. In particular, Santus et al. (2014) introduce the SLQS measure, which computes a transformed distributional space by weighting contexts via their overall entropy (Shannon, 1948). For any given two terms u and v, the final measure de-

pends on distribution of their top-k entropy values, reasoning that the narrower term should have less entropy in its contexts. Follow up studies found the SLQS measure is sensitive to hyperparameter tuning (namely, selection of k, the number of contexts to consider), but outperforms other measures with careful tuning (Shwartz et al., 2017).

Evaluating Unsupervised Measures

We motivate our work with an original comparison of the available measures based on the DIH (Equations 3.2–3.6) on their ability to separate hypernymy from three other relations: co-hyponymy, meronymy, and random. We also include cosine similarity as a scientific control. We evaluate using the BLESS dataset (Baroni and Lenci, 2011), which contains annotations of word relations for 200 mostly unambiguous, concrete nouns like 'van', 'horse', and 'chair'. These concepts are divided into 17 general Categories, like 'vehicle', 'ground mammal' and 'furniture'. Each noun is annotated with its co-hyponyms, meronyms, hypernyms and some random pairs. The original dataset additionally contains some additional verb relations, but we use only the noun-noun relationships, totally about 14k data points.

We compute the similarity between the LHS and RHS for each pair in BLESS using each of the measures. We then compute the Average Precision for each concept and relation types, and reported the Mean Average Precision (MAP) across all 200 concepts. Table 3.1 shows these MAP scores for the different relations and measures. An ideal *entails* measure would have a MAP of 1.0 in the Hyper column, and 0.0 in other columns.

Overall, we find none of the measures give an actively higher score for the hypernymy relation than the other three relations, indicating a dramatic failure of all the chosen measures. Indeed, two of the measures even (slightly) underperform cosine similarity. This raises the question: are the measures just overly sensitive to noise in distributional vectors, or is the Distributional Inclusion Hypothesis fundamentally flawed? If the unsupervised are measures are simply too sensitive to noise, perhaps using supervised techniques can improve performance. To this end, we shift

Measure	Co-hyp	Hyper	Mero	Random
cosine	.68	.20	.27	.27
balAPinc	.56	.23	.31	.28
WeedsPrec	.52	.22	.33	.28
Clarke	.66	.19	.28	.28
invCL	.60	.18	.31	.28

Table 3.1: Mean Average Precision (MAP) for the unsupervised entailment measures on the BLESS dataset. An ideal measure has a MAP of 1.0 in the Hyper column and 0.0 in the other columns. All measures fail to select hypernymy with greater precision than co-hyponymy.

our attention to *supervised* methods, hypothesizing that they will be more robust to the noise of distributional vectors.

3.3 Importance of Experimental Conditions

As we turn our attention to supervised methods , we first consider how experimental setup can significantly impact performance and yield misleading results.

As with all supervised machine learning methods, it is important to create distinct *training* and *testing* splits of the data. The training split is used to learn any model parameters, while the testing split is used to evaluate a model's generalization by checking that the model is able to make predictions on *unseen* data. Traditional methods for creating train/test splits of data are simple randomization (80% for train, 20% for test, stochastically assigning each data item to one or the other). When a particular dataset is somewhat small, it is often common to evaluate using cross-validation (CV), where the data is split into k equal sized *folds* and then each fold is used as the test set in k separate evaluations. Typically the performance metric (like accuracy) is averaged across all k folds. When the k in cross-validation is taken to be the full dataset size, then we arrive at a Leave-one-out cross-validation (LOOCV) experimental setup.

As with other datasets, we can evaluate our hypernymy datasets using any of these setups. For example, we could randomly assign 80% of the word pairs in our dataset as training, and utilize the remaining 20% as testing. However, due

the construction of the different datasets, many words appear multiple times in the dataset: sometimes they appear on the LHS, sometimes on the RHS, and always annotated with several example relationships. This means if we separate our training and testing data using standard random splits, then sometimes particular words will appear both in training and in test splits, and we may have issue generalizing to new pairs containing unseen words. Table 3.2a demonstrates this issue in an example, toy dataset. Crucially, we see that in this toy example, three words appear in both the training and testing sets.

To remedy this, we propose a variation of LOOCV where an individual *concept* is withheld as a particular test set, and the training set consists of all the remaining pairs which do not contain any lexical overlap. For example, if (cat, animal) appears in our test set, then the test set will consist of all (cat, *) pairs, and all (*, animal) pairs will be discarded from the training set. This adversarial experimental setup ensures that we can truly ensure our model is able to identify the lexical relationship of novel pairs, and not simply memorizing certain words. Table 3.2b demonstrates our setup.

Unfortunately, one downside of this setup is that some pairs must be discarded in each particular train/test setup, as shown in the Discarded section of Table 3.2b. This will occur in any dataset where a particular word occurs in more than one example pair. This can be mitigated by keeping the size of each test set to its smallest reasonable value to minimize the possibility of overlap with other pairs. Figure 3.3 shows the percentage of training data that must be discarded on two different datasets. A traditional machine learning setup, where no pairs are discarded because overlap is disregarded, would lie at the $y = 0.0$ line. We see that the amount of data that must be discarded rapidly increases with the size of the test set due to the increased vocabulary coverage; indeed, in a basic ten-fold CV setting, we need to discard 83% of the BLESS training data, and in three-fold CV we must discard 99%. We conclude that using many small folds is better for data efficiency, though this also comes at a computational cost, since we must run our classifiers for many more trials.

A third evaluation setting can also exist for the BLESS dataset, which con-

LHS	RHS	Category	Label
Test set			
cat	animal	mammal	hyper
sofa	tooth	furniture	rand
Training set			
dog	animal	mammal	hyper
dog	cat	mammal	co-hyp
dog	book	mammal	rand
cat	desk	mammal	rand
sofa	chair	furniture	hyper

(a) random split

LHS	RHS	Category	Label
Test set			
cat	animal	mammal	hyper
cat	desk	mammal	rand
Training set			
dog	book	mammal	rand
sofa	chair	furniture	hyper
sofa	tooth	furniture	rand
Discarded			
dog	animal	mammal	hyper
dog	cat	mammal	co-hyp

(b) our split

Table 3.2: (a) An example randomized train/test set split for a toy dataset. Note how the words 'cat', 'animal', and 'sofa' appear in both training and testing sets. (b) Our chosen splitting of the same data. Notice no words appear in both training and in test, but some pairs will necessarily be discarded.

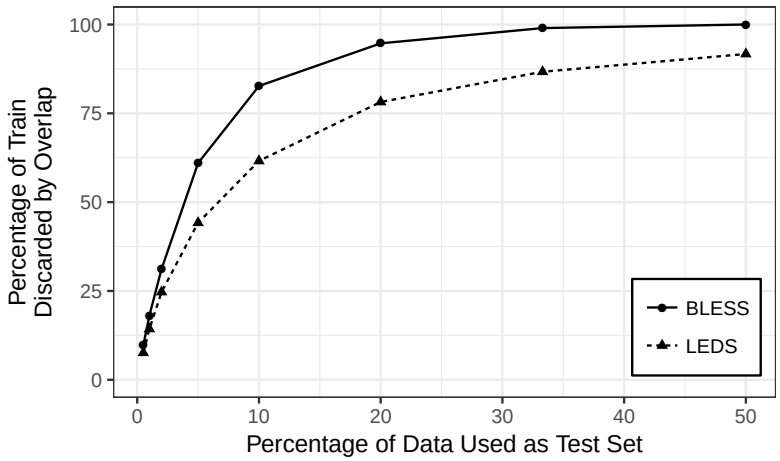


Figure 3.3: The amount of training data that must be eliminated by removing pairs sharing words in common between training and test, as a function of test set size. As the test set size increased, the number of overlapping pairs increases rapidly.

Stratification	Training Examples	W/ overlap	W/o overlap
Random	13,464	.976	-
Concept (LHS)	13,358	.971	.765
Category	12,332	.710	.652

Table 3.3: Comparing experimental conditions with and without lexical overlap on the classifier proposed by Baroni et al. (2012).

tains additional annotations of *Category* along with each concept. These 17 high-level categories include 'ground mammal', 'fruit', 'amphibian/reptile', 'furniture' and 'musical instrument'. Thus, we may also consider leave-out-one-*Category* CV.

We compare these three evaluation settings (traditional, LOO Concept, and LOO Category) by training and evaluating a fixed, existing supervised model. Namely, we use the model of Baroni et al. (2012), who use an off-the-shelf SVM with a polynomial kernel, reasoning that a polynomial SVM is able to capture the importance of features and their interactions (Cortes and Vapnik, 1995). Following their work, we use the concatenation of the raw vectors of a dimensionality-reduced, small-window, bag-of-words distributional space for input features. That is, for our running example, we use the features,

$$\text{features}(\mathbf{v}, \mathbf{u}) = [\vec{\text{cat}}; \vec{\text{animal}}].$$

Table 3.3 shows the accuracy of this classifier with these features on the four-way classification task of BLESS, both with lexical overlap and without lexical overlap. We did not evaluate a random split setting without lexical overlap.

One clear pattern from these results is a substantial decrease in performance when switching from allowing lexical overlap to removing lexical overlap: the concept-level stratification performance drops by over 20 points, and the category stratification drops by nearly 6 points. This substantial decrease shows one of our major observations: that some classifiers may be prone to a special form of over-fitting, where random pairs like ('cactus', 'animal') are classified as hypernymy, because the model simply *memorizes* that 'animal' is always a hypernym when it appears on the RHS, or that 'cactus' is always a hyponym when it appears on the

LHS.

This discovery of memorization was simultaneously published by both Roller et al. (2014) and Weeds et al. (2014), but each chose to deal with the issue in separate ways. The former controlled for memorization using the concept-level stratification we describe above, while the latter controlled for by comparing models with a randomized control variant, which used *random* vectors to represent each word. Models which had little difference between their real and randomized results were assumed to be memorizing the data, while models which improved over random vectors were considered strong. This is a very different approach to controlling for lexical memorization. We consider this setup to be inadequate, since randomized vectors are unable to exploit the distributional hypothesis, and therefore systematically underestimate the robustness of some models.

Our second observation is that stratification by category is substantially more difficult than stratification by concept, regardless of whether or not lexical overlap is allowed. The reason for this is threefold: (1) Stratification by category already accounts for a large percentage over lexical overlap, as many of the overlapping pairs are drawn from the same high-level categories. (2) Since there are only 17 categories, but 200 concepts, this moves us from 200-fold CV to 17-fold CV, causing our training set sizes to be significantly smaller (~99% vs ~88% of the full data). This is exacerbated when disallowing lexical overlap, as it moves us significantly further along the curve shown in Figure 3.3. (3) Category-level stratification changes the problem into a sort of domain-transfer setting, where we must learn about lexical relationships in the domain of mammals and then generalize to relationships about furniture, thus making the task markedly harder, as one must consider that the co-occurrences of 'cat' and 'stool' will likely overlap very little.

Clearly, the choice of experimental setup yields vastly different pictures of how well a model generalizes to new data. Evaluating with zero lexical overlap is important if we wish to distinguish between a model which truly generalizes to novel words and one which simply memorizes the data. Category stratification is important when we are interested in generalizing models to new domains; unfortunately, category information is only available in one dataset, making it difficult to

test robustness of models across multiple datasets.

On the other hand, Shwartz et al. (2016) argue that allowing lexical overlap is possibly more realistic, since lexicographers are more likely to want to insert words into an existing and incomplete ontology, rather than induce entirely new ontologies from scratch. In our own work, we hope to learn about what kinds of entailment signals are available in distributional vectors, and what these signals tell us about lexical relationships, rather than absolute performance on ontology completion. Therefore, we consider concept-level stratification with zero lexical overlap to be most appropriate for our own research. Since the models we consider in this chapter are computationally inexpensive, we employ a Leave-out-one-Concept setup for the remainder of this chapter.

Our proposal to evaluate settings with zero lexical overlap between training and test has been highly influential, and has since been adopted by numerous other publications since (Levy et al., 2015b; Kruszewski et al., 2015; Roller and Erk, 2016b; Shwartz et al., 2016; Vylomova et al., 2016). Therefore we consider this one of the core contributions of our thesis. Although in our own work we use either an n-fold (Roller and Erk, 2016b) or LOOCV setup (Roller et al., 2014), others have instead used either used fixed train/val/test sets with zero lexical overlap (Levy et al., 2015b; Shwartz et al., 2016; Vylomova et al., 2016) or allow for lexical overlap only with negatively labeled pairs (Kruszewski et al., 2015). We will briefly reconsider these alternatives in the next chapter.

3.4 Asym

We now introduce Asym (Roller et al., 2014), our first model for predicting hypernymy using distributional vectors, and another one of the major contributions of this thesis. At its core, the model is inspired by the famous result of Mikolov et al. (2013), who observed that vector subtraction can be used to perform some kinds of analogical reasoning in some kinds of distributional spaces:

$$\text{king} - \text{man} + \text{woman} \approx \text{queen}.$$

Interestingly, this vector subtraction approach reasonably models many grammatical relationships (singular/plural, verb conjugations) and some limited semantic relationships (gender, capital/country). Our proposed Asym model exploits this behavior for lexical relationship prediction.

The Asym model is a simple model which uses the *vector difference* between the hypothesized hypernym-hyponym pair as input features to an off-the-shelf classifier. For example, given a (unit normalized) distributional vector for 'animal' and a vector for 'cat', we use the vector $\vec{\text{animal}} - \vec{\text{cat}}$ as a positive example, while the vectors for $\vec{\text{cat}} - \vec{\text{animal}}$ and $\vec{\text{animal}} - \vec{\text{sofa}}$ are used as negative examples. Additionally, we also give the *element-wise squared difference vector* as features to the classifier. Formally, for a given (hypernym, hyponym) pair of words, $(\mathbf{h}, \mathbf{w})$, we compute the final feature space defined as:

$$\begin{aligned} a_i(\mathbf{h}, \mathbf{w}) &= \frac{\mathbf{h}_i}{\|\mathbf{h}\|} - \frac{\mathbf{w}_i}{\|\mathbf{w}\|} \\ b_i(\mathbf{h}, \mathbf{w}) &= a_i^2 \\ \text{features}(\mathbf{h}, \mathbf{w}) &= [\mathbf{a}; \mathbf{b}]\,, \end{aligned} \tag{3.7}$$

where $[a; b]$ is the vector *concatenation*. This computation is performed for all examples in our dataset, and then the features$(\mathbf{h}, \mathbf{w})$ vector and the classification label are used to train a Logistic Regression classifier.

3.4.1 Experiments and Results

We now compare our Asym model with the baseline model discussed above (Baroni et al., 2012), and show that Asym generalizes better on in settings with unseen words. As before, we evaluate on the BLESS dataset in a four-way classification task and measure performance in accuracy. We evaluate on Leave-One-Out Concept, so roughly 200-fold cross validation with zero lexical overlap.

We also evaluate using the Lexical Entailment Data (LEDS) (Baroni et al., 2012). This dataset consists of 2,770 word pairs, balanced between positive (housebuilding) and negative (leader-rider) examples of hypernymy, with 1376 unique

hyponyms (LHS) and 1016 unique hypernyms (RHS). The positive examples were generated by selecting direct hypernym relationships from WordNet, the negative examples by randomly permuting the hypernyms of the positive examples, and then manually checking correctness. As such, the LEDS name is somewhat misleading as it only contains hypernymy. Since it is a binary dataset, we train only a single binary classifier and evaluate performance in accuracy. As with BLESS, we use a LOO setting across the LHS with zero lexical overlap.

Since different types of distributional spaces exhibit different properties (Peirsman, 2008; Agirre et al., 2009; Baroni and Lenci, 2011), we first evaluate our model on two distributional spaces which use a simple Bag-of-Words context. The *Window-2 BoW* space counts content words two words to the left and right of targets as contexts, while the *Sentence BoW* space counts all content words within complete sentence boundaries. Both spaces were computed on a same concatenation of Wikipedia, ukWaC and BNC corpora, transformed using PPMI, and reduced to 300 dimensions using the SVD. We also use the precomputed TypeDM distributional space of Baroni and Lenci (2010), though this space was computed with different corpora and nonlinear transformations, making it not quite comparable to the other spaces.

We compare our model with two baselines: the first is a degenerate baseline, which guesses false for the (balanced) LEDS dataset, and always the most common label (*no-relation*) for BLESS. We also compare to the model proposed in Baroni et al. (2012) which was used in the above section on Experimental Setup.

Table 3.4 shows the results for our distributional space experiment. First we notice that both models strongly outperform the degenerate baseline, indicating there is some successful learning in the models, even with the lexical overlap considerations. We also see that the Window-2 space performs better than the Sentence space throughout, indicating it is likely the task depends more heavily *functional* properties of words than *topical* properties of words. However, we notice that the TypeDM space does worse than the Window-2 space, but we attribute this to the choice of nonlinear transformation (LMI) based on results from the next chapter.

We also see that the Asym model outperforms the model proposed by Baroni

Classifier	Space	BLESS	LEDS
Always guess false/no relation	-	.466	.500
SVM (Baroni et al., 2012)	Window 2	.765	.815
Asym (Roller et al., 2014)	Window 2	**.837**	**.850**
SVM	Sentence	.735	.778
Asym	Sentence	.803	.824
SVM	TypeDM	-	.655
Asym	TypeDM	.817	**.850**

Table 3.4: Accuracy of Baroni et al. (2012) and Asym on BLESS and LEDS using different spaces for feature generation. The Baroni et al. (2012) did not converge on the BLESS dataset with TypeDM vectors.

	BLESS		LEDS	
	SVM	LogReg	SVM	LogReg
Baseline	.461		.500	
Raw Vectors	.765	-	.815	.664
Normed	.456	.712	.230	.712
Diffs	.722	.769	.621	.774
Normed, Diffs (Asym)	.456	**.837**	.254	**.850**

Table 3.5: Accuracy of the different feature types for each of the machine learning algorithms on BLESS and LEDS. The LogReg classifier did not converge when using raw vectors on BLESS. Some settings do worse than baseline as a side-effect of the cross validation folds.

et al. (2012) throughout, indicating our architecture has better generalization, and is better suited for the task of hypernymy detection.

We also wish to consider the effects of our model's preprocessing requirements, namely that vectors are *unit-normalized* (Equation 3.7), and use the *vector differences*. To this end, we also compare the SVM to Logistic Regression with four variations used for features: simple concatenation of raw vectors (the features used by Baroni et al. (2012)), simple concatenation of unit-normalized vectors, vector difference(-squared) of the unnormalized vectors, and vector difference(-squared) of normalized vectors (the features used by Asym).

Table 3.5 shows each of these settings on both datasets Curiously, we see that each the two factors has nearly opposite behaviors on the two classifiers: using difference vectors hurts the SVM classifier on both datasets, but helps the LogReg classifier on LEDS. We also see that unit-normalizing vectors substantially hurts the performance of the SVM classifier, while giving an improvement to LogReg on LEDS. Nonetheless, the the final Asym model substantially outperforms both models on both datasets.

3.4.2 Selective Distributional Hypothesis

In an effort to understand *why* Asym outperforms the prior work, we now perform experiments to qualitatively understand its successes. We inspected the (linear) hyperplane weights learned by the Asym model, and found that the difference features ($\mathbf{a}$ in Equation 3.7) tended to learn mostly positive weights, capturing the directional and asymmetric aspects of hypernymy, namely that the *direction* of 'cat' to 'animal' is indicative of hypernymy. This bears a resemblance to the Distributional Inclusion Hypothesis, namely that we are interested in the inclusion of 'cat' within 'animal', but only within specific aspects. This suggests that Asym may be measuring a form of *Selective* Distributional Inclusion.

We test this interpretation of Asym as measuring a form of *Selective* Distributional Inclusion. After training the model's parameters on the BLESS dataset, we compare the model's learned hyperplane to the *context vectors* obtained in the Singular Value Decomposition. We select the 500 features most similar to the model's hyperplane, and then extract a distributional space from the original PPMI matrix limited to only these context items. If our Selective Distributional Inclusion Hypothesis is true, we would expect these 500 dimensions to highly compliment existing similarity measures based on the Distributional Inclusion Hypothesis from Section 3.2.1. We note that we are directly comparing unsupervised measures with a supervised model, and so this should only be understood as an experiment about the *interpretation* of our model, not its absolute performance.

Table 3.6 shows the results of our Selective DIH experiment. As expected, all measures except for cosine assign higher MAP values to hypernyms than they

	Original Space				Selective Space			
Measure	Co-hyp	Hyper	Mero	Rand	Co-hyp	Hyper	Mero	Rand
cosine	.68	.20	.27	.27	.69	.20	.24	.28
balAPinc	.56	.23	.31	.28	.54	.35	.26	.28
WeedsPrec	.52	.22	.33	.28	.50	.38	.27	.29
Clarke	.66	.19	.28	.28	.55	.39	.24	.29
invCL	.60	.18	.31	.28	.42	**.58**	.24	.29

Table 3.6: Mean Average Precision for the unsupervised measures before after selecting the top dimensions from the Asym model.

did in the original space, though only invCL that ranks hypernyms significantly higher than co-hyponyms.[1] We also see that the performance of our cosine baseline remains relatively unchanged by the feature selection procedure, and that the Clarke and invCL measures have their co-hyponymy and meronymy scores weakened. Altogether, this is evidence that the Asym measure is conforming to our Selective Distributional Inclusion interpretation.

3.4.3 Limitations of the Linear Model

We also inspect Asym's learned weights on the difference-squared features part of the hyperplane. Here we find that, opposite of the difference features, Asym mostly learns *negative weights*. Since all of the squaring always results in positively-valued features, we conclude that the difference-squared features are mostly informative for identifying *non*-hypernymy relations. We now show the value of the difference-squared features via a theoretical analysis.

After the publication of several supervised distributional models of hypernymy (Baroni and Lenci, 2011; Fu et al., 2014; Roller et al., 2014; Weeds et al., 2014), another study followed questioning whether these models truly learn to predict relationships. Levy et al. (2015b) hypothesized that each of these models is learning about *prototypicality*, or simply what a prototypical hypernym looks like. For example, after learning that 'cat is an animal' and that "dog is an animal," a

[1] Wilcoxon signed-rank test, $p < .001$

prototypicality classifier may also conclude that "sofa is an animal." That is, a prototypicality classifier will simply learn that *animal* is usually a hypernym, and will always predict this way.

The crux of the argument is explained analytically by Levy et al. (2015b), and hinges on observing that many of the models from the literature use *linear* classifiers (including Asym, but excluding the SVM in the previous experiments). Thus, consider a classifier which takes the concatenation of the vectors $[\mathbf{h}; \mathbf{w}]$ as features, and learns a hyperplane $\hat{\mathbf{p}}$ to make its prediction. Then the hyperplane $\hat{\mathbf{p}}$ can also be viewed as a concatenation of two vectors, $[\hat{\mathbf{h}}; \hat{\mathbf{w}}]$. Thus, the decision plane for a particular example $[\mathbf{h}; \mathbf{w}]$ can be analyzed as:

$$\begin{aligned} \hat{\mathbf{p}}^\top [\mathbf{h}; \mathbf{w}] &= [\hat{\mathbf{h}}; \hat{\mathbf{w}}]^\top [\mathbf{h}; \mathbf{w}] \\ &= \hat{\mathbf{h}}^\top \mathbf{h} + \hat{\mathbf{w}}^\top \mathbf{w} \\ &= \cos(\hat{\mathbf{h}}, \mathbf{h}) + \cos(\hat{\mathbf{w}}, \mathbf{w}) \end{aligned} \tag{3.8}$$

Note that the last step depends unit-normalizing the vectors, making the inner product the same as cosine similarity. This analysis by Levy et al. (2015b) shows that when the hyperplane $\hat{\mathbf{p}}$ is evaluated on a novel pair, it lacks any form of direct interaction between $\mathbf{h}$ and $\mathbf{w}$ like the inner product $\mathbf{h}^\top \mathbf{w}$, but rather only learns to capture the notion of hypernymy through $\hat{\mathbf{h}}$ and $\hat{\mathbf{w}}$, the *prototypicality vectors*. Without having some form of interaction, this Concat classifier has no way of estimating the relationship between the two words. Furthermore, a linear classifier which uses only the difference vectors will also have this flaw:

$$\begin{aligned} \hat{\mathbf{p}}^\top (\mathbf{h} - \mathbf{w}) &= (\hat{\mathbf{h}} - \hat{\mathbf{w}})^\top (\mathbf{h} - \mathbf{w}) \\ &= \hat{\mathbf{h}}^\top \mathbf{h} + \hat{\mathbf{w}}^\top \mathbf{w} - \hat{\mathbf{h}}^\top \mathbf{w} - \hat{\mathbf{w}}^\top \mathbf{h} \\ &= \cos(\hat{\mathbf{h}}, \mathbf{h}) + \cos(\hat{\mathbf{w}}, \mathbf{w}) - \cos(\hat{\mathbf{h}}, \mathbf{w}) - \cos(\hat{\mathbf{w}}, \mathbf{h}) \end{aligned}$$

This difference-only classifier's behavior is slightly improved over the plain concatenation classifier, due to the fact that it also measures *dissimilarity* between the prototypical hypernym with the hyponym, and the dissimilarity between the hy-

pernym with the prototypical hyponym, but it still crucially lacks any direct inner product between h and w.

However, this difference-only classifier is still not the same as Asym, since Asym additionally has the difference-squared features. By the Law of Cosines, we see that the square-difference features contain the crucial inner product term:

$$\begin{aligned}
\sum_i (h_i - w_i)^2 &= \sum_i h_i^2 + w_i^2 - 2(h_i w_i) \\
&= \sum_i h_i^2 + \sum_i w_i^2 - 2\sum_i h_i w_i \\
&= \mathbf{h}^\top \mathbf{h} + \mathbf{w}^\top \mathbf{w} - 2\mathbf{h}^\top \mathbf{w} \\
&= \cos(\mathbf{h}, \mathbf{h}) + \cos(\mathbf{w}, \mathbf{w}) - \boxed{2\cos(\mathbf{h}, \mathbf{w})}
\end{aligned}$$

This analysis is consistent with our earlier observation that the hyperplane weights learned by Asym are always negative: since the difference-squared features measure dissimilarity, they should be indicative of *non*-hypernymy relationships. Alternatively, one may view the difference-squared features act as a weighted Euclidean distance, and the further $\mathbf{h}$ and $\mathbf{w}$ are from each other, the less likely they are exhibit a hypernymy relationship.

3.5 Chapter Summary

We considered the task of hypernymy detection using distributional vectors. We reviewed numerous unsupervised approaches to the problem and the prevalence of the Distributional Inclusion Hypothesis. After comparing these unsupervised measures on a modern dataset, we find they come up short and consider the task as a supervised machine learning problem.

We then consider one previous supervised approach and show that choice of experimental setup has a large impact on performance due to issues of lexical memorization, a special kind of overfitting which prevents lexical generalization.

We propose a novel experimental setup where we construct our training and test sets to have zero lexical overlap. We then propose Asym, a novel supervised hypernymy classifier and show that it outperforms the prior supervised classifier on two datasets. We consider how choice of distributional space and vector preprocessing affects Asym's performance. We consider how Asym may be measuring a form of Selective Distributional Inclusion, and provide evidence for this behavior via its effect on unsupervised Distributional Inclusion measures. Finally, we consider why Asym is successful in its predictions, and show it is immune to a critical flaw found in other linear models.

Chapter 4
Lexical Entailment Detection with H-features

This chapter shows a new model of lexical relationship predictions, and related material. The work in this chapter has been published in Roller and Erk (2016b), and all work in this chapter constitutes original contributions.

4.1 Chapter Overview

In the previous chapter, we considered one novel model for hypernymy detection, and showed how experimental setup can vastly affect one's conclusions about effectiveness of models on this task. Since then, most work has considered the importance of experimental setup and lexical overlap, and a plethora of supervised distributional approaches have been proposed for different variations of the task. Yet, much of the literature differs on which model is best (Weeds et al., 2014; Roller et al., 2014), or whether models are sufficiently powerful to learn beyond prototypicality judgments (Levy et al., 2015b). In this chapter, we examine *why* prototypicality models are successful at all, and consider alternative explanations beyond simple lexical memorization. Our observations cumulate in a new model of hypernymy detection that is both highly interpretable, and outperforms other models in the literature on multiple datasets. Additionally, our model can be generalized to model other non-hypernymy relationships.

4.2 Supervised Distributional Models and Prototypicality

As discussed earlier, the earliest supervised approach was that of Baroni et al. (2012), who used an off-the-shelf polynomial SVM train on the *concatenation* of the word pair. Such a model obtains strong performance on randomized stratifications of datasets, but performs poorly when testing for lexical generalization. This issue was observed simultaneously by Roller et al. (2014) and Weeds et al.

(2014), who published these concerns at the same conference, and proposed different experimental methods for controlling for lexical memorization. Both papers also proposed variations of supervised models which use the vector difference as a basis for features inputted to a classifier: Weeds et al. (2014) proposed Diff, which uses *only* the vector difference, while Roller et al. (2014) noted the importance of the additional difference-squared terms, as discussed in the previous chapter.

Interestingly, the two works arrive at differing conclusions about the best model for the task of supervised hypernymy detection. While Roller et al. (2014) find vector differences to be useful and outperform a concatenation based model (Concat), Weeds et al. (2014) find the opposite: a simple, linear model based on vector concatenation outperforms the vector difference model in their experiments. Later, Vylomova et al. (2016) examined the models further and found that vector differences is robust at predicting a variety of lexical relationships, both with and without lexical overlap, but Shwartz et al. (2016) report better results using simple concatenation.

Levy et al. (2015b) focus on the issue of lexical memorization, and propose that both the Diff and Concat models act as *prototypicality* models, which merely look to identify words that appear most like prototypical hypernyms or hyponyms. They point out that such prototypicality models lack explicit terms relating the hypernym and hyponym, and therefore will predict that pairs like (‘animal’, ‘cactus’) will be predicted as hypernymy, because ‘animal’ looks like a typical hypernym, and ‘cactus’ looks like a typical hyponym. This analysis suggests that simple linear models, like Concat or Diff, should be unable to learn the relationships between words, and therefore will predict a large number of false positives.

Some works have proposed nonlinear models which can address these prototypicality concerns. Levy et al. (2015b) proposed a custom SVM kernel *Ksim*, which explicitly includes the critical inner product missing in linear models. Kruszewski et al. (2015) focus more on the role of the Distributional Inclusion Hypothesis within the framework of supervised models, and propose a custom neural network for hypernymy detection, which projects the distributional space into latent Boolean features. Both Levy et al. (2015b) and Kruszewski et al. (2015) compare

their proposed models with a carefully tuned SVM with an RBF, and find mixed results. These nonlinear models address the prototypicality concerns of Levy et al. (2015b), as they contain analytical terms necessary for similarity reasoning.

Thus, the literature generally disagrees about whether concatenation or vector differences are better representations for classification. Furthermore, despite analysis which suggests linear models should fail at the task entirely, several works find they actually perform robustly, both in traditional and adversarial experimental setups (Roller et al., 2014; Weeds et al., 2014; Fu et al., 2014; Shwartz et al., 2016; Vylomova et al., 2016). In the next section, we aim to resolve this conflict. We will consider what aspects of lexical meaning prototypicality models are capturing, and why this information allows prototypicality models to perform hypernymy detection so well.

4.3 Understanding Distributional Hypernymy Detectors

We turn our attention to understanding the mechanics of prototypicality models and the aspects of lexical meaning they learn. To this end, we study Concat extensively, and propose an alternative view of prototypicality: rather than simply modeling the "average" hypernym or hyponym, Concat actually acts as a kind of feature extractor which detects *aspects* typical of hypernyms. We study these properties by focusing only on binary classification experiments.

We focus on four different datasets, two of which are hypernymy-only datasets, and the remaining two are general lexical entailment datasets. The first two datasets are **LEDS** and **BLESS**, which contain both positive and negative examples of hypernymy, as discussed in the previous chapter. However, here we explicitly binarize the BLESS dataset such that all hypernymy examples in BLESS are labeled as positive, while random and other relations are labeled as negative. LEDS only contains hypernymy and non-hypernymy examples, and so is already binary.

The next dataset we consider is **Medical** (Levy et al., 2014a). This dataset contains noisy annotations of subject-verb-object entailments extracted from medical texts, and transformed into noun-noun entailments by argument alignments.

The data contains 12,600 annotations, but only 945 positive examples encompassing various relations like hypernymy, meronymy, synonymy, and some examples of "contextonymy," or entailments that occur in *some* specific contexts, but do not cleanly fit in other categories (e.g. 'hospital' entails 'doctor'). It one of the most difficult datasets: it is both highly domain specific (containing mostly medical terms) and highly unbalanced.

The final dataset we consider is **TM14**, a variation on the SemEval 2012 Shared Task of identifying the degree to which word pairs exhibit various relations. These relationships include a small amount of hypernymy, but also many more uncommon relations (agent-object, cause-effect, time-activity, etc). Relationships were binarized into (non-)entailing pairs by Turney and Mohammad (2015). The dataset covers 2188 pairs, 1084 of which are entailing.

These two entailment datasets contain important differences, especially in contrast to the hypernymy datasets. Neither contains any randomly generated negative pairs, meaning general semantic similarity measures should be less useful; And both exhibit a variety of non-hypernymy relations, which are less strictly defined and more difficult to model.

4.3.1 Distributional Vectors

Before we fully analyze Concat, we wish to describe our choice of distributional space, which will have profound repercussions in the next experiment. Namely, we desire to explore whether syntactic spaces or bag-of-words spaces are better suited for lexical entailment models. In the previous section, we considered one syntactic space, but did not control for the corpus, vocabulary, or co-occurrence weighting when comparing the syntactic space to the bag-of-words spaces. We now briefly consider an experiments with all parameters of space construction controlled for, and only vary the choice of context.

We construct several distributional spaces over a corpus containing the concatenation of Gigaword, Wikipedia, BNC and ukWaC. We preprocess the corpus using Stanford CoreNLP 3.5.2 (Chen and Manning, 2014) for tokenization, lemmatization, POS-tagging and universal dependency parses. We compute distributional

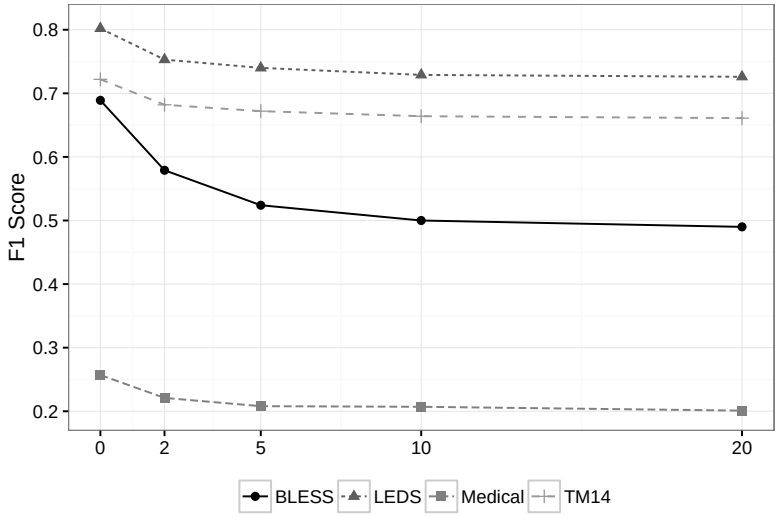


Figure 4.1: Comparison of different distributional window sizes for a baseline classifier on several different datasets. A window size of 0 contains the results for a syntactic distributional space.

vectors for 250k most frequent lemmas (with POS tags) by counting either their syntactic neighbors in a collapsed dependency parse, or a fixed bag-of-words window. Syntactic co-occurrences are limited to the one million most frequent contexts, while bag-of-words were limited to the 20k most frequent contexts. Co-occurrence counts were transformed using PPMI, and SVD reduced to 300 dimensions. Finally, all vectors were normalized to unit length. Altogether, this careful construction makes the vectors more comparable.

With all other factors held constant, we compare the effects chosen context using a baseline Concat model (Weeds et al., 2014) on the four datasets in an experimental setup comparable to the one in the previous chapter. Figure 4.1 shows the results comparing the choice of window size across all four datasets. The scores of the syntactic spaces are listed as a window size of 0.

Across all four datasets, we see that performance generally improves as we narrow the context window size, and improves dramatically with the selection of a syntactic distributional space. This indicates a strong preference for a kind of function similarity, as opposed to a topical similarity. Since the syntactic space out-

performs all bag-of-words spaces, we use this space for all experiments and models in the remainder of this chapter. Later, we will perform an analysis which explains the strong preference for syntactic spaces in all four datasets.

4.3.2 Reconsidering Prototypicality

As we saw in the previous chapter, the Concat model, which trains a linear classifier on the concatenation of word vectors $[\mathbf{h}; \mathbf{w}]$ has a fundamental flaw: that it does not explicitly model any direct interaction term between h and w. Rather, the hyperplane $\hat{\mathbf{p}}$ may be interpreted as learning a model of *prototypicality*, or how similar $\mathbf{h}$ is to the prototypical or average hypernym $\hat{\mathbf{h}}$. We show the analysis here again:

$$\begin{aligned} \hat{\mathbf{p}}^\top[\mathbf{h}; \mathbf{w}] &= [\hat{\mathbf{h}}; \hat{\mathbf{w}}]^\top[\mathbf{h}; \mathbf{w}] \\ &= \hat{\mathbf{h}}^\top\mathbf{h} + \hat{\mathbf{w}}^\top\mathbf{w} \\ &= \cos(\hat{\mathbf{h}}, \mathbf{h}) + \cos(\hat{\mathbf{w}}, \mathbf{w}) \end{aligned} \qquad \text{(3.8 revisited)}$$

As Levy et al. (2015b) point out, this analysis shows that Concat will fail on many randomly paired examples, like ('animal', 'cactus'), as $\vec{\text{animal}}$ is a rather prototypical hypernym, and $\vec{\text{cactus}}$ is a rather prototypical hyponym (of plant).

We agree with this prototypicality interpretation, although we believe it is incomplete: while it places a fundamental ceiling on the performance of these classifiers, it does not explain *why* others have found them to persist as strong baselines (Weeds et al., 2014; Roller et al., 2014; Kruszewski et al., 2015; Vylomova et al., 2016). To approach this question, we consider a baseline Concat classifier trained using a linear model. This classifier should most strongly exhibit the prototypicality behavior according to Equation 3.8, making it the best choice for analysis. We first consider the most pessimistic hypothesis: is it only learning to memorize which words are hypernyms at all?

We train the baseline Concat classifier using Logistic Regression on each of the four datasets, and extract the vocabulary words which are most similar to the $\hat{\mathbf{h}}$ half of the learned hyperplane $\hat{\mathbf{p}}$. If the classifier is only learning to mem-

LEDS	BLESS	Medical	TM14
material	goods	item	sensitiveness
structure	lifeform	unlockable	tactility
object	**item**	succor	palate
process	**equipment**	team-up	stiffness
activity	**herbivore**	non-essential	content

Table 4.1: Most similar words to the prototype $\hat{\mathbf{h}}$ learned by a Concat model. Bold items appear in the dataset. Very few of the closest terms directly appear in the dataset.

orize the training data, we would expect the most frequent hypernyms from the data to dominate this list of closest vocabulary terms, as the centroid $\hat{\mathbf{h}}$ should be strongly located near them. Table 4.1 gives the five words from the entire vocabulary, which are most similar to the learned hyperplane. Bold words appear directly in the dataset.

Interestingly, we notice there are very few bold words at all in the list. In LEDS, we actually see some good hypernyms of dataset items which do *not* even appear in the dataset. The Medical and TM14 words do not even appear related to the content of the datasets. Similar results were also found for Diff and Asym, and both when using Linear SVM and Logistic Regression. Lexical Memorization of frequent items cannot explain the success of the prototypicality classifiers in prior work. Instead, we propose an alternative interpretation of the hyperplane: that of a feature detector for hypernyms, or an *H-feature detector*.

4.3.3 H-Feature Detectors

Recall that distributional vectors are derived from a matrix M containing counts of how often words co-occur with the different syntactic contexts. This co-occurrence matrix is factorized using Singular Value Decomposition, producing both W, the ubiquitous word-embedding matrix, and C, the context-embedding matrix:

$$M \approx WC^\top \qquad \text{(2.2 revisited)}$$

LEDS	BLESS
nmod:such_as+animal	nmod:such_as+submarine
acl:relcl+identifiable	nmod:such_as+ship
nmod:of^{-1}+determine	nmod:such_as+seal
nmod:of^{-1}+categorisation	nmod:such_as+plane
compound+many	nmod:such_as+rack
nmod:such_as+pot	nmod:such_as+rope
Medical	**TM14**
nmod:such_as+patch	amod+desire
nmod:such_as+skin	amod+heighten
nmod:including+skin	nsubj^{-1}+disparate
nmod:such_as+tooth	nmod:such_as+honey
nmod:such_as+feather	nmod:with^{-1}+body
nmod:including+finger	nsubj^{-1}+unconstrained

Table 4.2: Most similar contexts to the prototype $\hat{\mathbf{h}}$ learned by the Concat model.

Since the word and context embeddings implicitly live in the same vector space (Melamud et al., 2015b), we can also compare Concat's hyperplane with the context matrix C. Under this interpretation, the Concat model does *not* learn what words are hypernyms, but rather what *contexts* are indicative of hypernymy. Table 4.2 shows the syntactic contexts with the highest cosine similarity to the $\hat{\mathbf{h}}$ prototype for each of the different datasets.

This view of Concat as an H-feature detector produces a radically different perspective on the classifier's hyperplane. Nearly all of the features learned take the form of Hearst patterns (Hearst, 1992; Snow et al., 2004). The most recognizable and common pattern learned is the 'such as' pattern, as in 'animals such as cats'. These patterns have been well known to be indicative of hypernymy for over two decades. Other interesting patterns are the 'including' pattern ('animals including cats') and 'many' pattern ('many animals'). Although we list only the six most similar context items for the datasets, we find similar contexts continue to dominate the list for the next 30-50 items.

Taken together, it is remarkable that the model identified these patterns using only distributional vectors and only the positive/negative example pairs. However, the reader should note these are not true Hearst patterns: Hearst patterns ex-

plicitly relate a hypernym and hyponym using an exact pattern match of a *single* co-occurrence. On the other hand, these *H-features* are *aggregate indicators* of hypernymy across a large corpus.

These learned features are much more interpretable than those found in the analysis of prior work like Roller et al. (2014) and Levy et al. (2015b). Roller et al. (2014) found no signals of H-features in their analysis of one classifier, but their model was focused on *bag-of-words* distributional vectors, which perform significantly worse on the task. Levy et al. (2015b) also performed an analysis of lexical entailment classifiers, and found weak signals like 'such' and 'of' appearing as prominent contexts in their classifier, giving an early hint of H-feature detectors, but not to such an overwhelming degree as we see in this work. Critically, their analysis focused on a classifier trained on high-dimensional, *sparse* vectors, rather than focusing on *context embeddings* as we do. By using sparse vectors, their model was unable to generalize across similar contexts. Additionally, their model did not make use of *collapsed dependencies*, making features like 'such' much weaker signals of entailment and therefore less dominant during analysis.

Among these remarkable lists, the LEDS and TM14 datasets stand out for having much fewer 'such as' patterns compared to BLESS and Medical. This is explained by the construction of the datasets: since LEDS contains the same words used as both positive and negative examples, the classifier has a hard time picking out clear signal. The TM14 dataset, however, does not contain any such negative examples.

We hypothesize the model cannot generalize because the TM14 dataset contains too many diverse and unrelated examples of lexical entailment, like instrument-goal (e.g. 'honey' $\rightarrow$ 'sweetness'). To test this, we retrained the model with only hypernymy as positive examples, and all other relations as negative. We find that 'such as' type patterns become top features, but also some interesting data specific features, like 'retailer of [clothes]'. Examining the data shows it contains many consumer goods, like 'beverage' or 'clothes', which explains these features.

4.4 Proposed Model

As we saw in the previous section, Concat acts as a sort of H-feature detector for whether h is a prototypical hypernym, but does not actually infer the relationship between h and w. Nonetheless, this is powerful behavior which should still be used in combination with the insights of other models like Ksim and Asym. To this end, we propose a novel model which exploits Concat's H-feature detector behavior, extends its modeling power, and adds two other types of evidence proposed in the literature: overall similarity, and distributional inclusion.

Our model works through an iterative procedure similar to Principal Component Analysis (PCA). Each iteration repeatedly trains a Concat classifier under the assumption that it acts as an H-feature detector, and then explicitly discards this information from the distributional vectors. By training a new H-feature detector on these modified distributional vectors, we can find *additional* features indicative of entailment which were missed by the first classifier. The entire procedure is iteratively repeated similar to how in Principal Component Analysis, the second principal component is computed after the first principal component has been removed from the data.

The main insight is that after training some H-feature detector using Concat, we can *remove* this prototype from the distributional vectors through the use of *vector projection*. Formally, the vector projection of $\mathbf{x}$ onto a vector $\hat{\mathbf{p}}$, $\text{proj}_{\hat{\mathbf{p}}}(\mathbf{x})$ finds the *component* of $\mathbf{x}$ which is in the direction of $\hat{\mathbf{p}}$,

$$\text{proj}_{\hat{\mathbf{p}}}(\mathbf{x}) = \left(\frac{\mathbf{x}^\top \hat{\mathbf{p}}}{\|\hat{\mathbf{p}}\|}\right) \hat{\mathbf{p}}.$$

Figure 4.2 gives a geometric illustration of the vector projection. If $\mathbf{x}$ forms the hypotenuse of a right triangle, $\text{proj}_{\hat{\mathbf{p}}}(\mathbf{x})$ forms a leg of the triangle. This also gives rise to the *vector rejection*, which is the vector forming the third leg of the triangle. The vector rejection is orthogonal to the projection, and intuitively, is the original

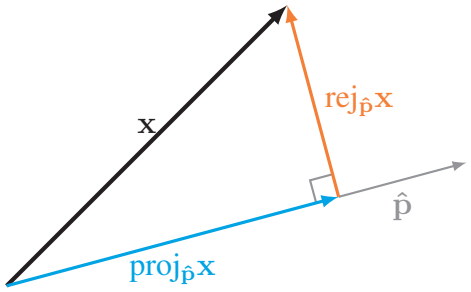


Figure 4.2: A vector $\hat{\mathbf{p}}$ is used to break $\mathbf{x}$ into two orthogonal components, its projection and the rejection over $\hat{\mathbf{p}}$.

vector after the projection has been removed:

$$\text{rej}_{\hat{\mathbf{p}}}(\mathbf{x}) = \mathbf{x} - \text{proj}_{\hat{\mathbf{p}}}(\mathbf{x}).$$

Using the vector rejection, we take a learned H-feature detector $\hat{\mathbf{p}}$, and discard these features from each of the word vectors. That is, for every data point $[\mathbf{h}; \mathbf{w}]$, we replace it by its vector rejection and rescale it to unit magnitude:

$$\mathbf{h}_{i+1} = \frac{\text{rej}_{\hat{\mathbf{p}}}(\mathbf{h})}{\|\text{rej}_{\hat{\mathbf{p}}}(\mathbf{h})\|}$$
$$\mathbf{w}_{i+1} = \frac{\text{rej}_{\hat{\mathbf{p}}}(\mathbf{w})}{\|\text{rej}_{\hat{\mathbf{p}}}(\mathbf{w})\|}$$

A new classifier trained on the $[\mathbf{h}_{i+1};\ \mathbf{w}_{i+1}]$ data *must* now learn a different decision plane than $\hat{\mathbf{p}}$, as $\hat{\mathbf{p}}$ is no longer present in any data points. This repetition of the procedure is roughly analogous to learning the second principal component of the data; we wish to classify the pairs without using any information learned from the previous iteration.

This second classifier *must* perform strictly worse than the original, otherwise the first classifier would have learned this second hyperplane. Nonetheless, it will be able to learn *new* H-feature detectors which the original classifier was unable to capture. By repeating this process, we can find several H-feature detectors, $\hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_n$. Although the first, $\hat{\mathbf{p}}_1$ is the best possible *single* H-feature detector, each

additional H-feature detector increases the model's representational power (albeit with diminishing returns).

This procedure alone does not address the main concern of Levy et al. (2015b): that these linear classifiers never actually model any connection between $\mathbf{h}$ and $\mathbf{w}$. To address this, we explicitly *compare* $\mathbf{h}$ and $\mathbf{w}$ by extracting additional information about how $\mathbf{h}$ and $\mathbf{w}$ interact with respect to each of the H-feature detectors. This additional information is then used to train one final classifier which makes the final prediction.

Concretely, in each iteration i of the procedure, we generate a four-valued feature vector $\mathbf{f}_i$, based on the H-feature detector $\hat{\mathbf{p}}_i$. Each feature vector contains (1) the similarity of $\mathbf{h}_i$ and $\mathbf{w}_i$ (before projection); (2) the feature $\hat{\mathbf{p}}_i$ applied to $\mathbf{h}_i$; (3) the H-feature detector $\hat{\mathbf{p}}_i$ applied to $\mathbf{w}_i$; and (4) the difference of 2 and 3.

$$\mathbf{f}_i(\mathbf{h}_i, \mathbf{w}_i, \hat{\mathbf{p}}_i) = \left[\underbrace{\mathbf{h}_i^\top \mathbf{w}_i}_{(1)} \; ; \; \underbrace{\mathbf{h}_i^\top \hat{\mathbf{p}}_i}_{(2)} \; ; \; \underbrace{\mathbf{w}_i^\top \hat{\mathbf{p}}_i}_{(3)} \; ; \; \underbrace{(\mathbf{h}_i - \mathbf{w}_i)^\top \hat{\mathbf{p}}_i}_{(4)} \right]$$

These four "meta"-features capture all the benefits of the H-feature detector (slots 2 and 3), while still addressing Concat's issues with similarity arguments (slot 1) *and* distributional inclusion (slot 4). The final feature's relation to the DIH comes from the observation of Roller et al. (2014) that the vector difference intuitively captures whether the hypernym *includes* the hyponym.

The concatenation of all the feature vectors $[\mathbf{f}_1; \ldots; \mathbf{f}_n]$ from repeated iteration form a $4n$-dimensional feature vector which we use as input to one final classifier which makes the ultimate decision. This classifier is trained on the same training data as each of the individual H-feature detectors, so our iterative procedure acts *only* as a method of feature extraction.

For our final classifier, we use a regularized SVM with an RBF-kernel, though decision trees and other nonlinear classifiers also perform reasonably well. The nonlinear final classifier can be understood as doing a form of logical reasoning about the four slots: 'animal' is a hypernym of 'cat' because (1) they are similar words where (2) 'animal' looks like a hypernym, but (3) 'cat' does not, and (4) some

'animal' contexts are not good 'cat' contexts. Additional iterations act as roughly like an "or" gate, so that separate H-features and inclusion patterns may be matched instead.

4.4.1 Experimental Setup and Evaluation

In our experiments, we use a variation of 20-fold cross validation which accounts for lexical overlap. This differs from the experimental setup in the previous chapter, which used leave-one-out cross validation. We choose this setup because computing the n prototype vectors considerably increases our computational costs, so we must reduce the number of folds to keep costs reasonable.

To simplify explanation of our setup, we first explain how we generate splits for training/testing, and then afterwards introduce validation methodology.

We first pool all the words from the antecedent (LHS) side of the data into a set, and split these lexical items into 20 distinct cross-validation folds, $D_1, \ldots, D_{20}$. For each fold D_i, we then use all pairs $[\mathbf{h}; \mathbf{w}]$ where $w \in D_i$ as the test set pairs. That is, if 'car' is in the test set fold, then 'car $\rightarrow$ vehicle' and 'car $\nrightarrow$ truck' will appear as test set pairs. The training set will then be *every pair* which does not contain *any* overlap with the test set; e.g. the training set will be all pairs which do not contain 'car', 'truck' or 'vehicle' as either the antecedent or consequent. This ensures that both (1) there is zero lexical overlap between training and testing and (2) every pair is used as an item in a test fold exactly once. One quirk of this setup is that all test sets are approximately the same size, but training sizes vary dramatically.

Our performance metric is F1 score. This is more representative than accuracy, as most of the datasets are heavily unbalanced. We report the mean F1 scores across all cross validation folds.

4.4.2 Hyperparameter Optimization

In order to handle hyperparameter selection, we actually generate the test set using fold i, and use fold $i-1$ as a validation set (removing pairs which would overlap with test), and the remaining 18 folds as training (removing pairs which would

Model	LEDS	BLESS	Medical	TM14
Linear Models				
Cosine	.787	.208	.168	.676
Concat	.794	.612	.218	.693
Diff	.805	.440	.195	.665
Asym	.865	.510	.210	.671
Concat+Diff	.801	.604	.224	.703
Concat+Asym	.843	**.631**	.240	.701
Nonlinear Models				
RBF	.779	.574	.215	.705
Ksim	.893	.488	.224	**.707**
H-feature Model	**.901**	**.631**	**.260**	.697

Table 4.3: Mean F1 scores for each model and dataset for binary classification.

overlap with test *or* validation). We select hyperparameters using grid search. For all models, we optimize over the regularization parameter $C \in \{10^{-4}, 10^{-3}, \ldots, 10^{4}\}$, and for our proposed model, the number of iterations $n \in \{1, \ldots, 6\}$. All other hyperparameters are left as defaults provided by Scikit-Learn (Pedregosa et al., 2011), except for using balanced class weights. Without balanced class weights, several of the baseline models learn degenerate functions (e.g. always guess non-entailing).

4.4.3 Results

We compare our proposed model to several existing and alternative baselines from the literature. Namely, we include a baseline Cosine classifier, which only learns a threshold which maximizes F1 score on the training set; three linear models of prior work, Concat, Diff and Asym; and the RBF and Ksim models found to be successful in Kruszewski et al. (2015) and Levy et al. (2015b). We also include two additional novel baselines, Concat+Diff and Concat+Asym, which add a notion of Distributional Inclusion into the Concat baseline, but are still linear models. We cannot include baselines like Ksim+Asym, because Ksim is based on a custom SVM kernel which is not amenable to combinations.

Table 4.3 the results across all four datasets for all of the listed models.

Model	LEDS	BLESS	Medical	TM14
No Similarity	.099	.061	.034	.003
No Detectors	-.008	.136	.018	.028
No Inclusion	.010	.031	.014	.001

Table 4.4: Absolute decrease in mean F1 on the development sets with the different feature types ablated. Higher numbers indicate greater feature importance.

Our proposed model improves significantly[1] over Concat in the LEDS, BLESS and Medical datasets, indicating the benefits of combining these aspects of similarity and distributional inclusion with the H-feature detectors of Concat. The Concat+Asym classifier also improves over the Concat baseline, further emphasizing these benefits. Our model performs approximately the same as Ksim on the LEDS and TM14 datasets (no significant difference), while significantly outperforming it on BLESS and Medical datasets.

4.4.4 Ablation Experiments

In order to evaluate how important each of the various f features are to the model, we also performed an ablation experiment where the classifier is *not* given the similarity (slot 1), prototype H-feature detectors (slots 2 and 3) or the inclusion features (slot 4). To evaluate the importance of these features, we fix the regularization parameter at $C = 1$, and train all ablated classifiers on each training fold with number of iterations $n = 1, \ldots, 6$. Table 4.4 shows the decrease (absolute difference) in performance between the full and ablated models on the development sets, so higher numbers indicate greater feature importance.

We find the similarity feature is extremely important in the LEDS, BLESS and Medical datasets, therefore reinforcing the findings of Levy et al. (2015b). The similarity feature is especially important in the LEDS and BLESS datasets, where negative examples include many random pairs. The detector features are moderately important for the Medical and TM14 datasets, and critically important on BLESS, where we found the strongest evidence of Hearst patterns in the H-feature detectors.

[1] Bootstrap test, $p < .01$.

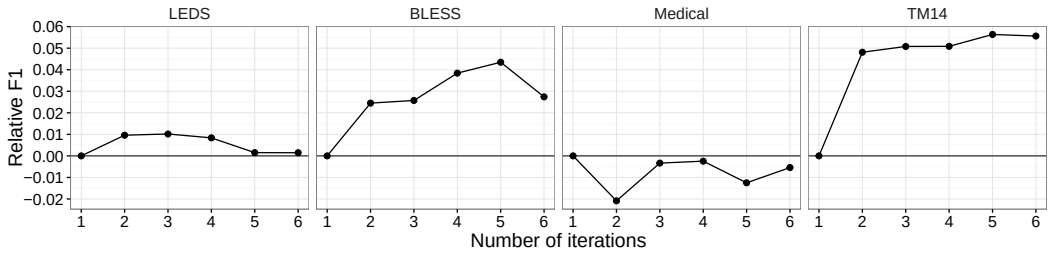


Figure 4.3: Performance of model on development folds by number of iterations. Plots show the improvement (absolute difference) in mean F1 over the model fixed at one iteration.

Surprisingly, the detector features are moderately *detrimental* on the LEDS dataset, though this can also be understood in the dataset's construction: since the negative examples are randomly shuffled positive examples, the *same* detector signal will appear in both positive and negative examples. Finally, we find the model performs somewhat robustly without the inclusion feature, but inclusion is still moderately impactful on three of the four datasets, lending further evidence to the Distributional Inclusion Hypothesis. In general, we find all three components are valuable sources of information for identifying hypernymy and lexical entailment.

4.4.5 Analysis by Number of Iterations

In order to evaluate how the iterative feature extraction affects model performance, we fix the regularization parameter at $C = 1$, and train our model fixing the number of iterations to $n = \{1, \ldots, 6\}$. We then measure the mean F1 score across the development folds and compare to a baseline which uses only one iteration. Figure 4.3 shows these results across all four datasets, with the 0 line set at performance of the $n = 1$ baseline. Models above 0 benefit from the additional iterations, while models below do not.

In the figure, we see that the iterative procedure moderately improves performance LEDS, while greatly improving the scores of BLESS and TM14, but on the medical dataset, additional iterations actually hurt performance. The differing

Iteration 1	Iteration 2
nmod:such_as+submarine	nmod:including+animal
nmod:such_as+ship	nmod:including+snail
nmod:such_as+seal	nmod:including+insect
nmod:such_as+plane	nmod:such_as+crustacean
nmod:such_as+rack	nmod:such_as+mollusc
nmod:such_as+rope	nmod:such_as+insect
Iteration 3	**Iteration 4**
amod+free-swimming	advcl+crown
nmod:including^{-1}+thing	advcl+victorious
nsubj^{-1}+scarcer	nsubj+eaters
nsubj^{-1}+pupate	nsubj+kaine
nmod:such_as+mollusc	nmod:at+finale
nmod:of^{-1}+value	nsubj+gowen

Table 4.5: Most similar contexts to the H-feature detector for each iteration of the PCA-like procedure. This model was trained on all data of BLESS. The first and second iterations contain clear Hearst patterns, while the third and fourth contain some data-specific and non-obvious signals.

curves indicate that the optimal number of iterations is very dataset specific, and provides differing amounts of improvement, and therefore should be tuned carefully. The LEDS and BLESS curves indicate a sort of "sweet spot" behavior, where further iterations degrade performance.

To gain some additional insight into what is captured by the various iterations of the feature extraction procedure, we repeat the procedure from Section 4.3.3: we train our model on the entire BLESS dataset using a fixed four iterations and regularization parameter. For each iteration, we compare its learned H-feature detector to the context embeddings, and report the most similar contexts for each iteration in Table 4.5.

The first iteration is identical to the one in Table 4.2, as expected. The second iteration includes many H-features not picked up by the first iteration, mostly those of the form '[animals] including [cats]'. The third iteration picks up some dataset specific signal, like 'free-swimming [animal]' and 'value of [computer]', and so on. By the fourth iteration, the features no longer exhibit any obvious Hearst patterns,

perhaps exceeding the sweet spot we observed in Figure 4.3. Nonetheless, we see how multiple iterations of the procedure allows our model to capture many more useful features than a single Concat classifier on its own.

4.5 Comparing to Path-based Entailment Detectors

We have thus far shown that our proposed model strongly outperforms other models for detecting hypernymy relations and lexical entailments using distributional semantics. However, as discussed in earlier chapters, there exists a wide literature in detecting lexical entailments directly using Hearst patterns (Hearst, 1992) or their Dependency path equivalents (Snow et al., 2004; Girju et al., 2006). These Hearst approaches are known to produce low recall, since words do not always appear with their hypernyms; yet Hearst approaches also may be complemented by distributional approaches. One recent line of work, called HypeNET, aims to obtain the best of both worlds by integrating both path-based and distributional-based methods together using modern neural network techniques.

For a given pair of words (h, w), HypeNET extracts the complete set of dependency paths connecting h and w in any sentence in a large corpus. These paths are then encoded into vectors using Long Short Term Memory networks (Hochreiter and Schmidhuber, 1997), and averaged together to obtain a final vector representing the potential Hearst patterns between h and w. This path vector is learned to maximize performance a set of binary hypernymy/non-hypernymy training pairs. The Path vector may be evaluated directly ("Path-based only"), or concatenated with distributional representations of h and w ("Hybrid Path and Dist."). Shwartz et al. (2016) find a Hybrid approach outperforms Path-only and Distributional-only comparisons.

We aim to directly compare with this approach by evaluating our model using the same data and experimental setup used by Shwartz et al. (2016). They use a custom dataset of hypernymy pairs extracted from a union of several datasets (including BLESS and DBPedia), and use only *related* words as negative examples, preventing simple cosine baselines from doing well. The full dataset contains about

	Random Split			No Overlap		
	P	**R**	**F1**	**P**	**R**	**F1**
Concat	.901	.637	.746	.754	.551	.637
Path-based only	.811	.716	.761	.691	.632	.660
Hybrid Path and Dist.	.913	**.890**	**.901**	**.809**	.617	.700

(a) results from Shwartz et al. (2016), reprinted here.

	Random Split			No Overlap		
	P	**R**	**F1**	**P**	**R**	**F1**
Concat	.910	.874	.892	.746	.949	**.836**
H-feature model	**.926**	.850	.886	.700	**.964**	.811

(b) our results

Table 4.6: Comparing the H-feature model with Shwartz et al. (2016), a state-of-the-art system which combines a path-based with a distributional approach.

70k pairs, roughly 25% of which are positively entailing.

Shwartz et al. (2016) evaluate in both a random-split and zero lexical overlap settings, arguing that both are important: random-split measures our ability to *complete* an existing taxonomy, while zero lexical overlap measures ability to *induce* a taxonomy. They use fixed 70% train, 5% validation, and 25% test folds, which they publish along with their paper. Table 4.6(a) shows results for a linear distributional-only baseline (Concat), along with their novel Path-only model and state-of-the-art Hybrid model. We emphasize these numbers are reprinted directly from Shwartz et al. (2016), and not produced by any of our own experiments.

To compare our H-feature model to theirs, we use their published datasets, and evaluate using the same training/test splits, and select hyperparameters using the same validation sets. Since not all words in the dataset appear in our set of distributional vectors, we assign out-of-vocabulary words the $\mathbf{0}$ vector. We evaluate using Concat, so the only difference between our baseline and theirs is the choice of distributional vectors. We also evaluate our H-feature model.

Table 4.6(a) shows the reported results from Shwartz et al. (2016), and Table 4.6(b) shows our results. We see that our Concat and H-feature models both

have a substantially higher F1 score than any of the models tested by Shwartz et al. (2016) in the zero lexical overlap setting, giving a new State-of-the-Art on this dataset. Since our Concat model does substantially better than any of their models, our improvements are likely due to the choice of distributional space. Shwartz et al. (2016) chose to use pretrained GloVe embeddings (Pennington et al., 2014), which use a fixed bag-of-words context window of 10. As we saw in Figure 4.1, a choice of syntactic distributional vectors substantially outperforms a bag-of-words distributional space. Interestingly, the Hybrid and Path-only models should encode the syntactic paths entirely, but these results suggest that, when training in zero lexical overlap settings, collapsed dependency embeddings already capture sufficient statistics.

We also see that the Hybrid model of Shwartz et al. (2016) moderately outperforms both of our models in the Random split setting. This indicates that the Hybrid approach is still better for taxonomy completion settings.

We also notice that our Concat model slightly outperforms the H-feature model in both random and zero overlap settings. We found the H-feature model chose to use only one iteration of feature extraction in hyperparameter validation, possibly explaining why the Concat model slightly outperforms the H-feature model. We manually inspected the H-features learned across several iterations and found only the first iteration contained interesting Hearst pattern type features (e.g., `such_as`). We suspect that this may be an artifact of the dataset's construction, which included noisy examples from several data sources, but no random negative pairs.

4.6 Detecting Multiple Relations

In the previous sections, we focused on how H-features may be used to detect lexical entailments, and models were trained with a binary positive and negative system. It is an interesting question whether H-features will also be useful for performing multi-relation classification, where labels may consist of several relations. On the one hand, the task may appear to be harder and Hearst patterns for non-hypernymy relations are much more difficult to identify (Girju et al., 2006). On the

other hand, the negative examples in the above sections constituted many kinds of relationships (random, co-hyponymy, meronymy) and distinguishing between these may overall improve performance. Furthermore, the relations discussed in the previous sections were *only* noun-noun relations; however, a variety of *other* interesting relations exist, like *object-attribute* (noun-adjective), or *event-object* (verb-noun).

Indeed, some other works have already found some evidence that generalizing to non-hypernymy relations is possible with existing models from the literature. Vylomova et al. (2016) finds that a Diff model (Weeds et al., 2004) is able to successfully distinguish between many lexical relations, including hypernymy, meronymy, gender ('king/queen'), object-location ('fish/aquarium'), and action-object ('zip/coat'). They also find that additional negative sampling (via pair randomization) can improve performance, and that difference vectors are robust across several models of distributional spaces. Additionally, Shwartz and Dagan (2016a; 2016b) apply the hybrid path-and-distributional neural net (Shwartz et al., 2016) to multi-relation datasets and find that hybrid approaches generally outperform only-path or only-distributional approaches, but that the improvements are less pronounced than in hypernymy-only datasets.

4.6.1 Adapting H-models for Multi-relation settings

We now consider the necessary modifications to extend our H-feature to multi-relation settings, or apply it to relationships which contain more diverse settings.

In Section 4.4, we proposed iteratively training multiple H-feature extractors through the use of vector rejection. During each iteration of feature, we learned a single H-feature detector $\hat{\mathbf{p}}$ which maximized predictive accuracy. However, each classifier has two parts, $\hat{\mathbf{h}}$ and $\hat{\mathbf{w}}$, both of which are viable candidates as H-features. Since we wished to positively identify hypernyms, we always selected $\hat{\mathbf{h}}$ as our H-feature detector.

With multi-class settings, the choice of which hyperplane is muddied: for some relations, the choice between $\hat{\mathbf{h}}$ and $\hat{\mathbf{w}}$ may not be as obvious: perhaps some H-features are better learned from the RHS. Additionally, since we are performing

multi-class prediction, any linear model will necessarily produce many different hyperplanes. In the one-versus-rest training setting we opt for, we will learn n hyperplanes for each of the n possible relation labels. With these two choices combined, each iteration will have $2n$ possible H-feature choices.

We considered several options for how to select among the hyperplanes available for H-features, and selected a simple heuristic which appeared to consistently deliver quality H-features for several iterations: at each iteration, we select the hyperplane with the largest vector magnitude. Intuitively, this heuristic selects the most *pronounced* H-feature detector at each setting, and we find this heuristic works well in practice.

Covering multiple relations additionally requires that we handle some non-noun lexical items. Since datasets are not explicitly annotated with part-of-speech tags, but our word vectors are, it is not obvious when we see a word like 'show' whether we should select 'show/NN' or 'show/VB'. We address this by always selecting the vector corresponding to a word's *most common* POS tag. For example, since 'show/VB' is more common in our corpus than 'show/NN', we always select 'show/VB' as our representation for 'show'.

4.6.2 Data and Experiments

We evaluate our multi-relation H-feature detector on several existing datasets covering a wide variety of relations.

The first dataset we use is **BLESS** (Baroni and Lenci, 2011), which was also used in the single-class experiments; previously, we only used the Hypernymy labels as positive classes, and all other labels as negative, and limited the dataset to only its noun-noun pairs. Here, we include all relations with their positive labels, including the *attribute* and *event* relations which were excluded in previous settings. The attribute relation includes nouns with their basic adjectives or attributes, e.g. 'fast' is an attribute of 'train'. The event relation includes nouns with action verbs they participate as either the subject or object in, e.g. 'grenade destroys' or 'cook broccoli'. The inclusion of these relations results in a dataset which includes nouns, adjectives, and verbs, but adjectives and verbs each appear only in one relation type.

The second dataset we use is **EVAL** (Santus et al., 2015), which contains a wide variety of lexical relations: *hypernymy*, *antonymy*, *synonymy*, *has* ('map' has 'information'), *property* ('volcano' has the property 'hot'), and *material* ('clothes' are made of 'cotton'). The full dataset has 7,378 annotated pairs. Notably, EVAL lacks any random negative pairs, meaning that all pairs contain words semantically related in some way.

The third dataset we use is **Root9** (Santus et al., 2016), which contains 12,763 pairs consisting of *random*, *hypernymy*, and *co-hyponymy* relations. The dataset was intentionally constructed to balance random relations with non-random relations, and balance the non-random relations equally between hypernymy and co-hyponymy.

The final dataset we use is **K&H+N** (Necsulescu et al., 2015), our largest dataset containing of 57,510 annotated pairs consisting of *hypernymy*, *co-hyponymy*, *meronymy*, and *random* relations and covering three domains (animals, plants, and vehicles). Similar to BLESS and LEDS, this dataset was extracted from WordNet, and was constructed to emphasize identifying relations between words which *do not co-occur* in the BNC. The K&H+N name derives from the dataset introduced by Kozareva and Hovy (2010), and extended by Necsulescu et al. (2015) to include co-hyponymy and meronymy. The resulting dataset is 90% either random or co-hyponymy, but still contains over 1,000 meronymy examples and 4,000 hypernymy examples.

The distribution of each of the relations of each of the four datasets may be found in Table 4.7.

4.6.3 Experiments

We follow the same experimental setup described in Section 4.4.1. Datasets are divided into a 20-fold Cross Validation setting with zero lexical overlap, with rotating validation and testing folds. We report mean *accuracy* across all folds, as F1 is poorly suited for non-binary datasets. We use balanced class weights for both H-feature extraction, and training the final meta-classifier, and tune the regularization parameter in $C = \{10^{-4}, \ldots, 10^{4}\}$ and number of iterations $N = \{1, \ldots, 10\}$ using

EVAL		BLESS		Root9		K&H+N	
Hypernym	25.5%	Hypernym	5.0%	Hypernym	25.0%	Hypernym	7.5%
Antonym	21.7%	Co-hyp	13.4%	Co-hyp	25.1%	Co-hyp	44.9%
Synonym	14.7%	Meronym	11.1%	Random	49.9%	Meronym	1.8%
Has A	7.4%	Attribute	10.3%			Random	45.9%
Material	4.3%	Event	14.4%				
Property	17.6%	Random	45.8%				
Part Of	8.9%						

Table 4.7: The makeup of each of the four multi-relation datasets, by relation type.

Model	EVAL	BLESS	Root9	K&H
Linear Models				
Concat	.469	.665	.581	.716
Diff	.442	.600	.514	.697
Asym	.453	.602	.671	.718
Concat+Diff	.459	.690	.596	.708
Concat+Asym	.464	.688	.644	.709
Nonlinear Models				
RBF	.455	.625	.560	.739
Ksim	.459	.633	.657	.744
H-feature Model	**.470**	**.713**	**.732**	**.755**

Table 4.8: Mean Accuracy scores for each model and dataset for multi-relation classification.

grid search. We found it necessary to increase the maximum number of iterations to 10 (from 6 in the binary experiments), as the larger datasets and more diverse relations required additional H-features.

We compare our model with the same baselines and prior work as done in our own binary experiments. We note that many of these baselines have never been applied to these datasets before.

Table 4.8 shows the results of our experiments in multi-relation classification. We see that our H-feature model has the strongest performance in all four datasets. Our model only slightly (non-significantly) improves over the Concat baseline in the EVAL dataset. In the remaining three datasets, our H-feature model

improves over the next-strongest model by a significant margin (McNemar test, $p < .01$). Clearly, the H-feature model is more adept at identifying lexical relations than our baselines.

We may also analyze the H-features extracted by each of the iterations of the model. We train an H-feature model on each of the four datasets with 10 iterations of H-feature extraction. We compared each of these 40 H-features with their most similar contexts, and manually filtered the list to the 8 most interesting H-feature sets. We observed several hypernymy H-feature sets similar to those in Table 4.2, and excluded those from our consideration. Table 4.9 lists our 8 H-features, along with the dataset each was extracted from, which iteration it was learned in, and which relation it provides evidence for. In general, we found the H-features of the BLESS dataset to be higher quality and more interesting than other datasets, but we report at least one H-feature from each.

In general, we find the H-features in the multi-relation settings are strong identifiers of their corresponding relations. Many are not as obvious or as clear cut as the original Hypernymy H-features, but all are interesting nonetheless. The BLESS and Root9 co-hyponymy detectors identify some interesting patterns from genetics, like 'cross between [tomato] and [tobacco]' and '[cat] vaguely like a [dog]'. The K&H meronymy H-features identifies clear possessive constructions ('stallion's [tail]') and mirror the meronymy patterns proposed by Girju et al. (2006), though the contexts are strongly biased towards by its animal-heavy construction. The EVAL Material detector identifies strong material patterns ('[bookshelves] handcrafted from [wood]'), and the EVAL Has detector identifies simple prepositional phrases ('[garden] with one [flower]'). Altogether, we see some strong evidence that the H-features truly are indicative of the relations they are meant to identify, indicating our proposed model works strongly in multi-relation settings.

Some of the H-features are more prototypical than the others, and their most similar words (rather than contexts) tend to appear directly in the data. For example, the BLESS Meronymy H-feature identifies many features that are common of animal parts, like 'splayed [feet]' or 'swept back [wings]'. The BLESS Attribute H-feature learned to simply memorize color adjectives, which were some of the most

common attributes; as such, its prototypical contexts are mostly concrete nouns appearing explicitly with color modifiers The BLESS Event hyperplane appears only to segment verbs from the rest of the data, with contexts like 'chance to [eat]' or 'enough to [eat]'. This is unsurprising, as this was the only relation with any examples of verbs. Although these contexts tend toward prototypicality, it is also clear why they should be useful for the various relationships.

4.7 Other Works in Distributional Relationship Detection

As a final aside, we briefly describe a few other works in distributional relationship detection which are outside the scope of this chapter's narrative, but which should also be mentioned.

In this chapter, we focused predominantly on works which treat hypernymy detection as a binary or multi-class prediction problem, where the system is provided a pair of words and must predict the relationship between the two words. Other works have considered whether it is possible to learn a *regression* from hyponyms to hypernyms. In particular, Fu et al. (2014) attempt the regression using vector differences, but find that separate regression models must be trained for separate regions of distributional space found via k-means clustering. Espinosa Anke et al. (2016) find the clustering step better corresponds to domain-specificity, and propose learning separate regressions per-domain rather than per-cluster, showing substantial improvement. Nayak (2015) attempts to circumvent the domain specificity issue entirely using deep feed-forward neural networks, and find moderate improvements in precision over a baseline linear model.

4.8 Chapter Summary

In this chapter, we proposed a new model for detecting lexical relationships using distributional vectors. We showed that the choice of bag-of-words or syntactic embeddings has a large impact on performance of a baseline Concat model. We then showed that the Concat model may be interpreted by comparing its hyperplane in

the embedded *context* space. We found this interpretation gives direct evidence as an Hearst-like feature extractor, explaining Concat's high performance.

We propose our H-feature model, which exploits this behavior through repeated iterations of feature extraction in a PCA-like procedure, and learns a final classifier which obtains state-of-the-art performance on several datasets. We also compare our model to a state-of-the-art Hybrid path-and-distributional model, and found our model substantially outperforms the Hybrid model in an experimental setting with zero lexical overlap. Finally, we show the necessary extensions to our model in order to use it for multi-relation classification settings. We found our model outperforms baselines and the prior work on four datasets, and that the model learns interesting H-features corresponding to a variety of lexical relations.

BLESS: Co-hyponymy (Itr. 2)	**BLESS: Meronymy (Itr. 3)**
nmod:between^{-1}+cross	amod+splayed
nmod:like^{-1}+vaguely	amod+swept-back
amod+trusty	amod+low-set
nmod:between^{-1}+hybrid	dobj^{-1}+protrude
compound+striped	amod+upswept
nmod:like^{-1}+pull	nmod:with^{-1}+distinctive
BLESS: Attribute (Itr. 4)	**BLESS: Event (Itr. 6)**
amod^{-1}+broom	nmod:to^{-1}+chance
amod^{-1}+cutlass	acl:relcl^{-1}+fowl
amod^{-1}+pail	nsubj+weta
amod^{-1}+muff	acl:relcl^{-1}+hen
amod^{-1}+paintbrush	nmod:to^{-1}+enough
amod^{-1}+aga	nmod:to^{-1}+reason
EVAL: Material (Itr. 2)	**EVAL: Has A (Itr. 3)**
nmod:from^{-1}+handcraft	nmod:with^{-1}+one
nmod:of^{-1}+fashion	nmod:below^{-1}+have
nmod:out_of^{-1}+construct	nmod:above^{-1}+have
nmod:like^{-1}+material	nmod:with^{-1}+long
nmod:out_of^{-1}+fashion	nmod:with^{-1}+long
appos+fiber	nmod:with^{-1}+larger
K&H: Meronymy (Itr. 3)	**Root9: Co-hyponymy (Itr. 1)**
nmod:poss+stallion	nmod:as^{-1}+big
nmod:poss+serpent	nmod:between^{-1}+cross
nmod:with-1+beast	nmod:such_as^{-1}+variety
nmod:poss+lion	nmod:like^{-1}+pull
nmod:poss+hog	amod+trusty
nmod:poss+lizard	nmod:between^{-1}+hybrid

Table 4.9: Feature detectors for different relations learned by our model in a variety of datasets. These 8 hyperplanes were selected as most interesting among 40. Each hyperplane is listed with its corresponding dataset, identifying relation, and which iteration it was extracted from. Several Hypernymy hyperplanes were also observed, but are not shown here.

Chapter 5
Lexical Substitution

This chapter shows new model of lexical substitution, and related material. The work in this chapter has been published in Roller and Erk (2016a), and all work in this chapter constitutes original contributions.

5.1 Chapter Introduction

Our discussions in the previous chapters assumed that a word has only one meaning, and that this meaning does not change with respect to context. However, some lexical entailments may hold in particular contexts but not in others. In this chapter, we consider the task of *Lexical Substitution*, where we must propose paraphrases for a given target word appearing in a particular context. We propose Probability in Context (PIC), an unsupervised measure for estimating a paraphrase's appropriateness for a given target and context. We show our measure outperforms baseline measures, especially in an experimental setup where paraphrases are considered from the entire vocabulary.

5.2 Lexical Substitution

In our discussion of lexical relationships in the previous two chapters, we assumed that words are *monosemous*, or that they have only one meaning. However, words like 'bright' have multiple meanings, which change depending on context, as in 'bright girl' and 'bright coat'. When a word has multiple meanings, it is *polysemous*, and polysemy is yet another source of ambiguity in natural language.

It is still unclear what representation should even be chosen to represent polysemous words. One option is the use of a lexicon, like the dictionary. Lexicographers are responsible for cataloging and defining word senses; one may see this in the multiple definitions provided by the Oxford English Dictionary or Merriam

Webster, but even professional lexicographers often disagree on how many senses a word has. For example, 'bank' has three noun senses in Webster's dictionary, but only two in Oxford's. WordNet (Miller, 1995) is one of the most popular and heavily used computational dictionaries, and is notoriously fine-grained (it gives 10 noun senses for 'bank').

Even after one fixes the inventory of word senses, labeling a particular instance of a word with its sense is fraught with its own difficulties: non-experts have very low inter annotator agreements (Ng et al., 1999), and even lexicographers will disagree often (Kilgarriff and Rosenzweig, 2000). Results in computational approaches to Word Sense Induction (WSI) and Word Sense Disambiguation (WSD) reflect this task difficulty (McCarthy, 2009; Navigli, 2009).

Distributional Semantics offers an alluring alternative path, given its ability to measure the *graded* levels of similarity between words (Erk and Padó, 2008). One possibility to represent each of a word's senses using a separate vector, resulting in multiple vectors per word (Reisinger and Mooney, 2010; Huang et al., 2012), and then use these prototypes appropriately in downstream tasks. If one takes this idea to its absolute extreme, one can imagine a new vector for every instantiation of a word. In this extreme, one may choose not to create a unique vector for every word, but instead model how meaning *shifts* in a particular given context (Erk and Padó, 2008; Erk, 2010).

One way to measure this has been the Lexical Substitution task. In the Lexical Substitution task, we are provided with a sentential context, and must suggest *substitutes* which can replace the given target word, while preserving the meaning of the entire sentence (McCarthy and Navigli, 2007; Biemann, 2012; Kremer et al., 2014). For example, when provided the sentence

'The **bright** scientist reads the paper,'

annotators will readily offer substitutes like the 'clever scientist' or 'smart scientist'. Similarly, when provided the sentence

'The girl put on her **bright** coat,'

humans will suggest the 'colorful coat' instead. However, annotators may also choose creative or nonstandard suggestions, especially when provided excerpts from fiction. In one dataset, humans provided with the context

> 'Tara stood stock-still, waiting for the first tiny gleam from the scout craft to appear in the darkness of the **wormhole**,'

human annotators considered *portal* and *rift* to be excellent substitutes. These substitutes take into account the Science Fiction context of the sentence, indicating the task is more complicated than simple synonymy. Indeed, in one one lexical substitution dataset, only 9% of substitutes in their dataset were direct synonyms in WordNet (Kremer et al., 2014).

5.3 Prior Work

The original SemEval shared task released a moderately sized dataset and defined the task (McCarthy and Navigli, 2007). The experimental setup was specifically designed in order to elicit responses to polysemy in human annotators. The corpus collectors selected a set of 2211 sentences containing a targeted list polysemous words curated both manually and automatically using lexical resources. These words cover several part-of-speech tags, and sentences were manually selected to prevent primary senses from dominating the corpus. Five human annotators were then asked to propose substitutes for the target words in the sentential context: phrases and moderate generalizations were allowed, but discouraged. The unification of substitutions resulted in a ranked list of substitutes for every sentential context. Shared task participants were asked to perform the same task computationally: both suggesting and ranking possible substitutes.

Performance for the task was measured in *precision out of ten* (oot), where a system was able to make up to ten guesses and the average percentage of correct guesses across all sentences was reported; and *best*, where a system provided its single best substitute and evaluated on whether that substitute was proposed by any annotator.

A clear majority of systems used some lexical resource, like WordNet, to first provide a list of possible substitutions, and then filtered or ranked that list using their own research methods. Several used a mixture of co-occurrences or n-grams to find good matches, and several used distributional methods to rank the substitutes, almost no systems used existing WSD datasets (McCarthy and Navigli, 2007). The strongest team on the *best* metric used a supervised approach where substitutes were ranked using features like n-gram likelihood (Yuret, 2007), while the strongest performing system in the *oot* metric accidentally exploited a flaw in the evaluation by proposing the same substitute many times (Giuliano et al., 2007). The literature has since varied substantially on experimental setup and evaluation metrics. Nonetheless, the task provided an interesting test bed for a variety approaches, including distributional approaches, language modeling approaches, and some supervised approaches (Szarvas et al., 2013a).

5.3.1 Unsupervised approaches

One major insight from the competition was the importance of syntagmatic fit in lexical substitution with the success of n-gram based models. Erk and Padó (2008) proposed a model for offering each unique word occurrence its own individualized word vector, allowing polysemy to be modeled without explicit word senses, and applied these sense-free representations to the task of lexical substitution. Their representations modified the out-of-context word vectors by averaging it with other possible fillers for the same contexts. For example, in 'the fielder *catches*', they would consider verbs which take 'fielder' in the subject position. Since their work focused more on innovations in representations of polysemy, they evaluated their system on its ability to identify correct substitutes from a candidate list, and did not consider its ability to *generate* substitutes from the entire vocabulary.

Thater et al. (2009, 2010) expanded on the this idea of inverse selection preferences by proposing a *second-order* distributional space, where syntactic co-occurrences were modeled via a sparse tensor representation similar to TypeDM (Baroni and Lenci, 2011), and using multiple folds over the tensor to produce a

unique vector for each word in context. Thater et al. (2011) refine and simplify on this second-order view of the previous models by proposing to re-weight a word's syntactic co-occurrences based on their similarity to the present context. This unsupervised method remained one of strongest performing systems even when applied to new datasets (Kremer et al., 2014) and compared to some supervised systems. Along a similar path, Melamud et al. (2015a) proposed an alternative second order model based on using Google's web corpus of 5-grams.

Dinu and Lapata (2010) showed that latent-factor distributional models like Latent Dirichlet Allocation (Blei et al., 2003) could allow for evaluating the particular fit of a given substitute. They considered two latent factorization models (LDA and NNMF) and showed that context could successfully modulate over varying latent senses of a particular word in a bag-of-words context. This insight in the power of low-rank factorizations later turns out to be valuable in later lines of work (Melamud et al., 2015b; Roller and Erk, 2016a). We defer discussion of these works until Section 5.4, due to their central role in this chapter.

Van de Cruys et al. (2011) observed that syntax-based distributional models tend to provide the strongest results, yet individual examples in lexical substitution data *should* require a wider understanding of the context, like in the 'wormhole' example provided earlier. They proposed a *joint* latent factorization model over both a wide bag-of-words co-occurrences with the functional, syntactic context. Although this line of work has not seen a large degree of follow up, the author of this dissertation hopes future research will reconsider this path.

More recently, some research has used neural language models for the task of lexical substitution. Kawakami and Dyer (2016) propose using word alignments from translation tasks in order to identify the different senses of a word in the hidden state of a bidirectional RNN. The intuition is that two tokens which translate into the same foreign token have the same underlying sense, while two tokens which translate into different foreign tokens must represent a polysemous term, an idea has a long history in the literature (Resnik and Yarowsky, 1999; Diab, 2003; Bannard and Callison-Burch, 2005). Since their model requires substantial aligned translation corpora, it is not directly comparable to the other models exploring the task. Mela-

mud et al. (2016) applied the same neural architecture, trained in an English-only language modeling task, but found it did not achieve state-of-the-art performance.

5.3.2 Supervised approaches

Another vein of research has treated Lexical Substitution as a truly supervised research problem, as opposed to the predominantly unsupervised or knowledge-transfer approaches listed above. For example, the original SemEval task tuned a threshold on likelihoods obtained from a language model (Yuret, 2007). Others have proposed more sophisticated architectures based on producing context features for the target token (Biemann, 2012), or a fully delexicalized classifier which learns over features extracted via a (target, substitute) pair (Szarvas et al., 2013a). Another approach from Szarvas et al. (2013b) notes that previous approaches emphasize the importance of ranking properly, and proposes to treat lexical substitution as a true ranking problem. This substantially outperforms the other ranking approaches, but the comparison is unfair due to differing levels of supervision.

5.4 Context Vectors for LexSub (Melamud et al. 2015b)

We now consider the unsupervised models of Melamud et al. (2015b), which are central to the contributions of our own work. Melamud et al. (2015b) introduce a simple, but high-performing method, based on the Syntactic Skip-Gram Negative Sampling model of Levy and Goldberg (2014a). As with other unsupervised models, it combines a measure of an out-of-context substitute's suitability, with a special in-context appropriateness measure (Erk and Padó, 2008). Unlike the many of the previous unsupervised models, it does not rely on computing large second-order vectors, and can in fact be used with off-the-shelf syntactic word2vec vectors.

As emphasized in Chapters 2 and 4, when creating low dimensional vectors, a high-dimensional co-occurrence matrix $\mathbf{M}$ is factorized or approximated with two

lower-rank matrices (Levy and Goldberg, 2014b):

$$\mathbf{M} = \mathbf{V}\mathbf{C}^\top, \qquad \text{(2.2 revisited)}$$

where $\mathbf{V}$ is the matrix of word vectors, and $\mathbf{C}$ is the matrix of *context vectors*. The primary insight of Melamud et al. (2015b) was that these context vectors can be used to evaluate the suitability of a novel substitute in a particular given context.

Suppose we have sentence like 'The very **bright** girl reads', and we want to evaluate which is a better substitute: 'smart' or 'colorful'. If we were to extract the syntactic contexts of the target 'bright', as in the previous chapter, we would end up with exactly two contexts: `advmod:very`, indicating our target is modified by 'very', and $\texttt{amod}^{-1}\texttt{:girl}$, indicating that our target modifies 'girl'. We may then use the context matrix $\mathbf{C}$ to look up the vectors for these two contexts, and judge their similarity to the possible substitutes.

Figure 5.1 provides an intuitive illustration of the idea: by looking at the context vector $\texttt{amod}^{-1}\texttt{:girl}$, we see that it is closer to the substitute 'smart' than the substitute 'colorful'. The context `advmod:very`, on the other hand, lies somewhere about halfway between the two substitutes, indicating both substitutes are equally appropriate for this context.

Formally, for a given target t, substitute s and context C, the final addCos score is defined as:

$$\text{addCos}(\mathbf{s}|\mathbf{t}, C) = \cos(\mathbf{s}, \mathbf{t}) + \sum_{c \in C} \cos(\mathbf{s}, \mathbf{c}). \qquad (5.1)$$

Note that the equation has two components: the first measures the *out-of-context* similarity between the target t and the substitute s using basic cosine similarity. The second half of the measure encodes the *in-context* appropriateness, by considering substitute with respect to each of the syntactic contexts.

This formulation is considerably simpler than many of the other models in the prior work, which require significant overhead for space creation or re-weighting of terms, but performs competitively with even more recent models. For

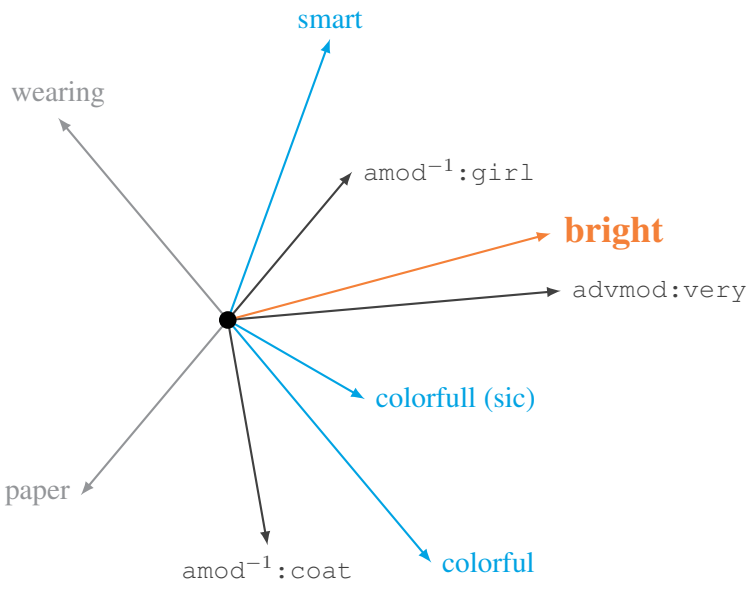


Figure 5.1: Cartoon illustration of how context vectors can disambiguate word senses.

this reason, it acts as a particular strong and standard baseline, which can help calibrate work with different vocabulary or experimental conditions.

Melamud et al. (2015b) also consider variants of this measure, including *balAddCos*, which equally weights the out-of-context similarity with the in-context appropriateness:

$$\text{balAddCos}(s|t, C) = \text{cosine}(s, t) + \frac{1}{|C|} \sum_{c \in C} \text{cosine}(s, c). \tag{5.2}$$

This alternative measure reweights the similarities obtained from the context vectors so that the out-of-context similarity and in-context appropriateness are given equal weight. The choice between the addCosand balAddCosis an empirical one, and performance of either could vary across datasets or evaluation metrics.

5.5 Probability-in-Context (PIC)

We propose an extension of the context vector measure of Melamud et al. (2015b), called Probability-in-Context (PIC). We note that our measure is *not* a well defined probability measure; rather the name was simply chosen for the purpose of the paper's title.

Similar to balAddCos, the measure has two equally-weighted, independent components measuring the appropriateness of the substitute for both the target and the context, each taking the form of a softmax:

$$\begin{aligned} \text{PIC}(s|t,C) &= P(s|t) \times P(s|C) \\ P(s|t) &= \frac{1}{Z_t} \exp\left\{\mathbf{s}^\top \mathbf{t}\right\} \\ P(s|C) &= \frac{1}{Z_C} \exp\left\{\sum_{c \in C} \mathbf{s}^\top \left[\mathbf{W}\mathbf{c} + \mathbf{b}\right]\right\}, \end{aligned} \tag{5.3}$$

where the t is the target word, s is a proposed substitute, C is the set of syntactic contexts, and Z_t and Z_C are normalizing constants. Our model differs from balAddCos in two important ways. First, our similarity measure uses *unnormalized* inner products $s^\top t$ and $s^\top c$, rather than *normalized* cosine similarities; and second, we introduce two free parameters, W and b, which act as a linear transformation over the original context vectors. These parameters are estimated from the *original corpus*, and are trained to maximize the prediction of a *target* from only its syntactic contexts. These parameters are *not* trained on the lexical substitution dataset, and therefore serve only to tune how the fixed distributional vectors act in this alternative objective function.

Nonetheless, to identify the contribution of this parameterization versus the softmax objective, we also introduce to a non-parameterized PIC (*nPIC*), which

does not contain the extra parameters:

$$\begin{aligned} \text{nPIC}(s|t, C) &= P(s|t) \times P_n(s|C) \\ P_n(s|C) &= \frac{1}{Z_n} \exp\left\{\sum_{c \in C} \mathbf{s}^\top \mathbf{c}\right\} \end{aligned} \tag{5.4}$$

One may ask why not train or tune the embeddings to optimize this softmax objective directly, rather than learning the parameterization which adjust the embeddings? This ultimately is an empirical question, and our initial tests found that training new embeddings from scratch did not result in stronger performance. Additionally, the SGNS embeddings are already popular and well-understood in relation to traditional distributional semantics. Furthermore, we can re-use the embeddings of Melamud et al. (2015b) to ensure that experimental conditions are fair to both models, and that the quality of embeddings remains consistent across comparisons.

5.6 Experiments

We compare our proposed measures to three baselines: OOC, the Out-of-Context cosine similarity between the word and target $\cos(s, t)$; and the addCos and balAddCos measures. We evaluate on three lexical substitution datasets:

- **SE**: The dataset used in the original SemEval 2007 shared task (McCarthy and Navigli, 2007) consists of 201 words manually chosen to exhibit polysemy, with 10 sentences per target. For a given target in a particular context, five annotators were asked to propose up to 3 substitutes. As all our experiments are unsupervised, we always evaluate over the entire dataset, rather than the original held-out test set.

- **Coinco**: The Concepts-in-Context dataset (Kremer et al., 2014) is a large lexical substitution corpus with proposed substitutes for nearly all content words in roughly 2,500 sentences from a mixture of genres (newswire, emails, and fiction). Crowdsourcing was used to obtain a minimum of 6 contextually-appropriate substitutes for over 15k tokens. Unlike SemEval, this dataset was

not constructed specifically to capture polysemy; nonetheless, many substitutes vary considerably for different sentential contexts, indicating that polysemy is definitely present.

- **TSWI**: The Turk bootstrap Word Sense Inventory 2.0 (Biemann, 2012) is a crowdsourced lexical substitution corpus focused on about 1,000 common English nouns. The dataset contains nearly 25,000 contextual uses of these nouns. Though the dataset was originally constructed to induce a word-sense lexicon based on common substitution patterns, here we only use it as a lexical substitution dataset.

We compare models on two variations of the lexical substitution task: candidate ranking and all-words ranking. In the *candidate ranking* task, the model is given a list of candidates and must select which are most appropriate for the given target. We follow prior work in pooling candidates from all substitutions for a given lemma and POS over all contexts, and measure performance using Generalized Average Precision (GAP) (Kishida, 2005). GAP is similar to Average Precision, but weighted by the number of times a substitute was given by annotators. Intuitively, it measures the number of "correct" points the model has obtained at every point, normalized by the maximum number of possible points obtainable at that rank. For a particular target, suppose $h_1, \ldots, h_n$ are an ordered list of the model's scores assigned for the possible substitutes, so that the substitute with the best score comes first. Let's also assume that $g_1, \ldots, g_n$ is the sorted gold list of scores, and that $g(h_i)$ gives the gold score of the substitute predicted at h_i. GAP is then defined in terms of the precision over all possible values of i:

$$\text{GAP} = \frac{1}{R} \sum_{k=1}^{n} \left[\sum_{i=1}^{k} \frac{g(h_i)}{i} \right]$$

$$R = \sum_{k=1}^{n} \left[\frac{1}{k} \sum_{i=1}^{n} g_i \right]$$

Intuitively, GAP measures 1.0 when the model perfectly ranks the list, and 0.0 when the model provides the reverse listing. Partial credit is awarded for providing items

in roughly the correct order, weighted by how important the items are in the gold scores.

Our second task is the much more difficult task of *all-words ranking*. In this task, the model is not provided any gold list of candidates, but must select possible substitutes from the entire vocabulary. All models are also hardcoded not to predict substitutes with the same stem as the target, e.g. for the 'bright girl' example, models cannot predict 'brighter' or 'brightest'. On this all-words ranking task, we evaluate performance with (micro) mean Precision@1 and P@3: that is, of a system's top one/three guesses, the percentage also given by human annotators. These evaluation metrics are similar to the *best* and *oot* metrics reported in the literature, but we find P@1 and P@3 easier to interpret and analyze. In case annotators did not provide three unique substitutes, that particular substitution is divided by the total number of unique substitutions.

5.6.1 Vectors and Training Procedure

We use the word and context vectors released by Melamud et al. (2015b), which were previously shown to perform strongly in lexical substitution tasks, and to ensure our model is directly comparable with theirs. These embeddings were computed from a corpus of (word, relation, context) tuples extracted from ukWaC and processed using the dependency-based word2vec model of Levy and Goldberg (2014a). These embeddings contain 600d vectors for 173k words and about 1M syntactic contexts.

To train the W and b parameters, we extract tokens with syntactic contexts using the same corpus (ukWaC), parser (Chen and Manning, 2014), and extraction procedure used to generate the embeddings. See Melamud et al. (2015b) for complete details. After extracting every token with its contexts, we randomly sample 10% of the data to reduce computation time, leaving us with 190M tokens for training W and b. We use sampled softmax to reduce training time (Jean et al., 2015), sampling 15 negative candidates uniformly from the vocabulary, optimizing cross-entropy over just these 16 words per sample. We optimize W and b in one epoch of stochastic gradient descent (SGD) with a learning rate of 0.01, momentum of

Measure	SE	CoInCo	TWSI
OOC	44.2	44.5	57.9
addCos	51.2	46.3	62.2
balAddCos	49.6	46.5	61.3
nPIC	51.3	46.4	61.8
PIC	**52.4**	**48.3**	**62.8**

Table 5.1: Performance of nPIC and PIC on the three Lexical Substitution datasets in the all candidate ranking task, measured in GAP.

Measure	SE	CoInCo	TWSI
OOC	11.7	10.9	9.8
addCos	12.9	10.5	7.9
balAddCos	13.4	11.8	9.8
nPIC	17.3	16.3	11.1
PIC	**19.7**	**18.2**	**13.7**

(a) Precision@1

Measure	SE	CoInCo	TWSI
OOC	9.7	8.6	7.0
addCos	9.0	7.9	6.1
balAddCos	9.8	9.1	7.4
nPIC	13.1	12.1	7.9
PIC	**14.8**	**13.8**	**10.1**

(b) Precision@3

Table 5.2: Performance of nPIC and PIC on the three Lexical Substitution datasets in the all words ranking task, measured in (a) P@1 and (b) P@3.

0.98, and a batch size of 2048. We found all of these hyperparameters worked well initially, and did not tune them. This procedure took around 45 minutes using a GPU.

5.6.2 Results

Table 5.1 compares the models on the substitute ranking task only. The first observation we make is that the PIC measure performs best in all evaluations on all datasets by a significant margin (Wilcoxon signed-rank test, $p < 0.01$). In these GAP evaluations, all measures perform substantially better than the OOC baseline, and the nPIC measure performs comparably to balAddCos. We note that context-sensitive measures give the most improvement in SemEval, reflecting its greater emphasis on polysemy.

As we turn to the all-words ranking evaluations in Table 5.2, we observe that

‘You can sort of challenge them well, did you **really** know the time when you said yes?’

OOC	balAddCos	nPIC	PIC
trully	proably	realy	**actually**
actually	trully	**truly**	**truly**
actaully	acutally	**actually**	already
acutally	actaully	hardly	barely
proably	probaly	**definitely**	just

Table 5.3: Example where the PIC performs better in the All-Words Ranking task. The target word and correct answers are bolded.

‘As a general rule, point of view should not **change** during a scene.’

OOC	balAddCos	nPIC	PIC
sea-change	**alter**	reoccur	re-occur
alter	sea-change	re-occur	appear
shift	**shift**	prevail	overstate
downshift	downshift	deviate	differ
re-configure	increase/decrease	divulged	disappear

Table 5.4: Example where the PIC performs worse the All-Words Ranking task. The target word and correct answers are bolded.

the absolute numbers are much lower, reflecting the increased difficulty of the task. We also see the that nPIC and PIC both improve greatly over all baselines: The nPIC measure is a relative 30% improvement over balAddCos in SE07 and Coinco, and the PIC measure is a relative 50% improvement over balAddCos in 5 evaluations. Indeed we find that PIC significantly outperforms other models (Wilcoxon signed-rank test, $p < 0.01$).

Since both measures have a clear improvement over the baselines, especially in the more difficult all-words task, we next strive to understand why.

5.6.3 Analysis

We first examine two hand-selected examples to provide intuitions about the strengths and weaknesses of our model. Table 5.3 contains a cherry-picked example where both our measures outperform prior work. OOC and balAddCos both suggest replacements with reasonable semantics, but contain misspelled substitutes. nPIC and PIC only pick words with the correct spellings, with the exception of 'realy'. This suggests our models are predicting terms with higher frequency.

Table 5.4 shows a lemon-picked example where our models perform strictly worse than the baseline models. We notice that the unusual 'sea-change' item is prominent in the OOC and balAddCos models, but has dropped from the rankings in our models, also indicating the model may be picking more frequent terms.

We consider a few experiments with this hypothesis that the measures do better because they capture better *unigram* statistics than the baselines. Recent literature found that the vector norm of SGNS embeddings correlates strongly with word frequency (Wilson and Schakel, 2015). We verified this for ourselves, computing the Spearman's rank correlation between the corpus unigram frequency and the vector length and found $\rho = 0.90$, indicating the two correlate very strongly. Since the dot product is also the unnormalized cosine, it follows that nPIC and PIC should depend on unigram frequency. One can view this intuitively in Figure 5.1: although 'colorfull' has a better cosine-similarity to the target than the correctly spelled 'colorful', we tend to prefer the more frequently chosen term since its vector has a higher magnitude.

To verify that the nPIC and PIC measures are indeed preferring more frequent substitutes, we compare the single best predictions (P@1) of the balAddCos and nPIC systems on all-words prediction on Coinco. Roughly 42% of the predictions made by the systems are identical, but of the remaining items, 74% of predictions made by nPIC have a higher corpus frequency than balAddCos (where chance is 50%). We find balAddCos and PIC make the same prediction 37% of the time, and PIC predicts a more frequent word in 83% of remaining items. The results for SE07 and TWSI2 are similar.

This indicates that the unigram bias is even higher for PIC than nPIC. To gain more insight, we manually inspect the learned parameters W and b. We find that the W matrix is nearly diagonal, with the values along the diagonal normally distributed around $\mu = 1.11$ ($\sigma = 0.02$) and the rest of the matrix normally distributed roughly around 0 ($\mu = 2 \times 10^{-5},\ \sigma = 0.02$). This is to say, the PIC model is approximately learning to *exaggerate* the magnitude of the dot product, $s^\top c$. This suggests one could even replace our parameter W with a single scaling parameter, though we leave this for future work.

To inspect the bias b, we compute the inner product of the b vector with the word embedding matrix, to find each word's a priori bias, and correlate it with word frequencies. We find $\rho = 0.25$, indicating that b is also capturing unigram statistics.

Is it helpful in lexical substitution to prefer more frequent substitutes? To test this, we pool all annotator responses for all contexts in Coinco, and find the number of times a substitute is given correlates strongly with frequency ($\rho = 0.54$).

These results emphasize the importance of incorporating unigram frequencies when attempting the lexical substitution task (as with many other tasks in NLP). Compared to cosine, the dot product in nPIC stresses unigram frequency, and the parameters W and b strengthen this tendency.

5.7 Chapter Summary

We have presented PIC, a simple new measure for assessing the appropriateness of a substitute in a particular context for the Lexical Substitution task. The measure assesses the fit of the substitute both to the target word and the sentence context using a combination of out-of-context similarity with in-context appropriateness. It significantly outperforms comparable baselines from prior work, and does not require any additional lexical resources. An analysis indicates its performance improvements derive from a tendency to lean more strongly on unigram statistics than baselines.

Chapter 6
Lexical Entailment in RTE

This chapter shows how the work of the previous three chapters may be combined into an end-to-end RTE system. Some of the contributions in this chapter are published in Beltagy et al. (2016). All material in this chapter constitutes original contributions, except as designated in Sections 6.2–6.3 and Section 6.4.1.

6.1 Chapter Overview

In the previous chapters, we showed how improvements in modeling can lead to better performance in several lexical tasks, including lexical relationship prediction, lexical entailment detection, and lexical substitution. Each of these tasks is valuable in its own right, but it is important not to lose sight of the bigger picture. We turn our attention now to the Recognizing Textual Entailment task (RTE), which we introduced in the beginning of Chapters 1 and 2.

In Recognizing Textual Entailment, a system is provided two sentences, a text (antecedent) and hypothesis (consequent), and must decide whether a human would say the hypothesis follows from the text. A particular RTE pair may involve different phenomena, but Dagan et al. (2006) observed that lexical entailments are often critical to reaching the proper conclusion. The following RTE example involves a several kinds of lexical entailment explored in this thesis, including hypernymy, polysemy, and event understanding:

> Text: The bright girl reads a book.
> Hypothesis: A smart child looks at a book.

Given the importance of lexical entailment in the RTE task, we expect that the contributions from previous chapters should be reflected in an end-to-end RTE system. We begin with a brief expository into the RTE system we use, and then integrate our contributions in directly.

6.2 RTE System

We use the RTE system of Beltagy et al. (2016) to evaluate our improvements in lexical entailment. Although this dissertation author is also a co-author on that paper, this section (6.2) primarily describes efforts of others, and should not be seen as a contribution of this thesis. In short, this section explains the RTE system at an abstract level, and considers how it reduces sentences into lexical entailment subproblems. We do not review other systems for RTE, as most approaches were outlined in Section 2.2, and their differences are not the focus of this document.

The system of Beltagy et al. (2016) performs textual entailment using *probabilistic* logical deduction. It computes a First Order Logical (FOL) representation of each sentence, and then estimates the probability that the second sentence is entailed using a probabilistic logic formalism called Markov Logic Networks (MLNs) (Richardson and Domingos, 2006). MLNs take as input a knowledge base of facts about the world in the form of weighted FOL formulas (including atomic formulas), and produces a graphical model which estimates the probability of formulas based on their consistency with the knowledge base. This powerful formulation is outside the scope of this document, but with careful consideration, MLNs are able to model a large variety of natural language phenomena, including quantifiers and negation (Beltagy, 2016).

In order to perform correct logical reasoning, the system must reduce the text and hypothesis into weighted logical formulas, and consider a database of relevant facts. To obtain a logical representation of the sentences, the RTE system employs Boxer (Bos, 2008), a wide coverage semantic analysis system, which converts syntactic parse trees into logical formulas.

The RTE system then aligns the representations of the text and hypothesis to find words and short phrases for which it needs lexical entailment judgments. This alignment procedure is done using a variant of Robinson Resolution, which identifies common predicates and performs unification across text and hypothesis.

Label	Antecedent/Consequent
Entailing	A: Two teams are competing in a football match C: Two groups of people are playing football
Contradicting	A: The brown horse is near a red barrel at the rodeo C: The brown horse is far from a red barrel at the rodeo

Table 6.1: Example entailing and contradicting sentences from the SICK dataset.

In our example, the system will produce two separate rules:

$$\text{bright girl} \rightarrow \text{smart child}$$

$$\text{read} \rightarrow \text{look at}$$

The RTE system then queries a lexical entailment system to predict whether these lexical items are entailed, neutral, or contradictory. These lexical entailment predictions are finally encoded as facts about the world, and complete probabilistic logical reasoning is performed about the full sentences (including quantification and negation) to come to a final prediction about sentential entailment.

6.3 RTE Data and Lexical Entailment Annotations

We evaluate using the Sentences Involving Compositional Knowledge (SICK) dataset, which contains nearly 10k sentence pairs, evenly split between training and test sets (Marelli et al., 2014). Sentence pairs were extracted randomly from image caption datasets, and then simplified and extended to cover semantic issues like negation, quantification, compositional language, and a variety of lexical entailment relationships like hypernymy and polysemy. Finally, sentences were manually annotated as *entailing* (the antecedent implies the consequent), *contradicting* (the antecedent implies the opposite of the consequent), or neutral (neither of the above). SICK's construction makes it an excellent dataset to test a complete RTE system, due to the rich variety of semantic phenomena it covers. Two examples from the dataset are shown in Table 6.1.

We saw above that we may break some sentences down into individual lexi-

cal entailments of short phrases. These lexical entailments may be either entailing, neutral, or contradicting. This differs from work in previous chapters, which never attempted to identify contradictions or reason about phrases. Additionally, lexical entailments in this chapter must be tailored specifically for the RTE task, and therefore require we build our own lexical entailment dataset.

To do this, we used the Robinson Resolution approach described above to extract all the lexical entailments possible in the system. Many of these lexical entailments could be automatically annotated from the RTE sentences themselves: if the sentence is entailing and contains no negation, it follows that the individual lexical entailments must also entail. This allows us to build a list of certainly true lexical entailments. Sentences that are contradicting and contain only a single lexical entailment rule can similarly have their lexical entailments automatically labeled as contradicting. Sentences which are neutral or contain certain logical phenomena may *not* be automatically annotated. Two authors of Beltagy et al. (2016) (including this dissertation's author) manually annotated lexical entailment pairs which could not be automatically annotated. We only annotated lexical entailment pairs derived from the training set, as we did not want pairs from the test set to possibly influence our decisions, and any lexical annotations derived from the test set would be considered cheating. The final lexical entailment dataset contains 10,213 annotated rules extracted from the SICK training set, and is made available for future research (Beltagy et al., 2016).

6.4 Lexical Entailment Classifier

We now turn our focus to the actual lexical entailment classifier used by the RTE system. This lexical entailment classifier uses a variety of techniques from prior work, as well as several novel contributions. As a starting point for the classifier, we employ a number of hand-engineered features shown by Lai and Hockenmaier (2014) to be useful in the SICK dataset. These features are given to a basic SVM classifier with an RBF kernel, which decides which of the three lexical entailment decisions to make. We also employ a greedy alignment procedure which

allows the lexical entailment classifier to classify short phrases as opposed to single words. Finally, we describe additional features for the lexical entailment classifier based on our findings about supervised distributional lexical entailment classifiers, and lexical substitution.

6.4.1 Baseline Features (Lai and Hockenmair (2014)

The baseline set of features we use were originally proposed by Lai and Hockenmaier (2014), and consist of basic phrase features (e.g. lengths of the phrases), wordform features (e.g. do these words carry the same lemma or POS?), WordNet features (e.g. are these two words hypernyms in WordNet), and distributional features (e.g. cosine similarity). A full list of these features may be found in Table 6.2, along with their types and counts.

The Wordform features include simple features which indicate whether both words have the same part-of-speech, lemma, and plurality. On their own, these features only capture the most simple entailments and slight prior information. The basic alignment features capture simple properties for the phrasal entailments, like the lengths of the LHS and RHS. These properties are generally semantically void, but do inform some entailment priors: for example, a single word is unlikely to entail a long phrase with modifiers or a prepositional attachment.

The Distributional features contain simple cosine similarities for the LHS and RHS using one syntactic distributional space and one bag-of-words distributional space. The syntactic distributional space is comparable to the one used in the previous sections, and was trained in roughly the same preprocessing as in Chapter 4: part-of-speech tagged, lemmatized, collapsed-dependencies, PPMI-transformed and SVD reduced to 300 dimensions. The BoW distributional space is a standard Word2Vec space, trained with a window size of 20 using the same preprocessed corpus as the syntactic space. We choose the large BoW window so that the dependency and BoW spaces would measure each extreme of the function-topic similarity spectrum.

Finally, the WordNet features are the most sophisticated, and contain simple extractions of WordNet relationships: whether the LHS and RHS are listed as

Name	Description	Type	#
Simple alignment Features			10
Length	Length of LHS, RHS and (absolute) difference	Real	4
Alignments	Number of (un)aligned words on LHS, RHS	Real	3
Align Pct.	Alignment statistics rescaled to percentages	Real	3
Wordform			18
Same word	Same lemma, surface form	Binary	2
POS	POS of LHS, POS of RHS, same POS	Binary	10
Sg/Pl	Whether LHS/RHS/both are singular/plural	Binary	6
Distributional features			28
OOV	True if either lemma not in dist space	Binary	2
BoW Cosine	Cosine between LHS and RHS in BoW space	Real	1
Dep Cosine	Cosine between LHS and RHS in syn. space	Real	1
BoW Hist	Bins of BoW Cosine	Binary	12
Dep Hist	Bins of Dep Cosine	Binary	12
WordNet			18
OOV	True if a lemma is not in WordNet	Binary	1
Hyper	True if LHS is hypernym of RHS	Binary	1
Hypo	True if RHS is hypernym of LHS	Binary	1
Syn	True if LHS and RHS is in same synset	Binary	1
Ant	True if LHS and RHS are antonyms	Binary	1
Path Sim	Path similarity (NLTK)	Real	1
Path Sim Hist	Bins of path similarity (NLTK)	Binary	12

Table 6.2: List of baseline features in the lexical entailment classifier, along with types and counts. Most of these were proposed by Lai and Hockenmaier (2014), with the exception of the Binning features.

synonyms, antonyms, or hypernyms of each other (directly or across a long chain) in WordNet. Additionally, the Path Similarity is also included, as implemented by Bird et al. (2009).

6.4.2 Alignment Features (Lai and Hockenmaier, 2014)

Since the lexical entailments that we are provided may possibly be short *phrases* rather than individual words, we also use a variation of the alignment procedure proposed by Lai and Hockenmaier (2014). In this alignment procedure, the

syntactic head of the phrases are assumed to be aligned, and the baseline features are computed between the heads of the LHS and RHS. The remaining words are then greedily aligned using distributional similarity: the distributional similarity between the every non-head word on the LHS and RHS is computed, and then the distributionally most similar pair is assume to be aligned and removed from the pool of unaligned words. This process is recursively repeated until one or both phrases has all its words exhausted. Finally, the hand-engineered features described in Table 6.2 are then computed for all the aligned pairs, and a min/mean/max of all the features across all aligned pairs is computed and used as additional features for the lexical entailment classifier. Since phrases may have been extracted from neutral sentences, sometimes the alignments may be very poor, but these bad alignments will be reflected in the statistics and learned as nonentailing by the classifier.

6.4.3 Binning of Similarity Features

The features and alignment procedure described above are primarily derived from the observations of (Lai and Hockenmaier, 2014), who won the original shared task for this dataset. However, we do include one novel addition to these features which was not proposed in prior work, which we call *similarity binning*. We extend the real-valued distributional-similarity features by *binning* the cosine similarities into discretized ranges (e.g. 0.0–0.1, . . . , 0.9–1.0), and transforming them into binary-valued indicator features. This stems from an observation that the probability of entailment is non-monotonic in distributional similarity.

We can observe this fact in Figure 6.1, which shows the distribution of entailment annotations as a stacked histogram over distributional similarities. Observe that mid-similar terms (those with a cosine of $\sim .80$, like *cat* and *animal*) are more likely entailments than those with high similarity (cosine of $\sim .95$, like *cat* and *dog*). We found this binning technique significantly improved the contribution distributional similarity in feature-engineered lexical entailment classifiers, and this binning represents a contribution of this thesis. We will visit the effect of binning in the Experiments section.

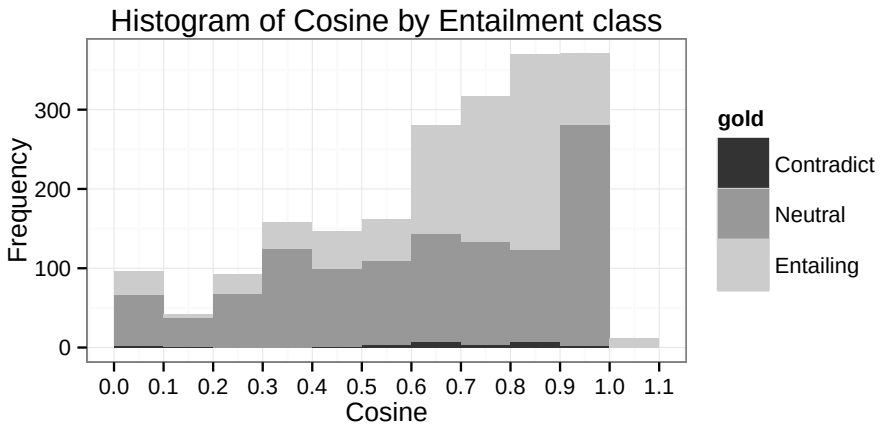


Figure 6.1: Stacked histogram showing the distribution of entailment relations on lexical pairs by cosine. Highly similar pairs (0.90–0.99) are less likely entailing than moderately similar pairs (0.70–0.89).

6.4.4 Distributional Classifiers

We also include several variations of features from the Supervised Distributional Lexical Relationship classifiers of Chapters 3 and 4. Namely, we consider and compare three main models:

- **Concat**: We include the vectors representing the head word of the LHS phrase and the head word of the RHS phrase, unit-normalized. This corresponds to the RBF model from Table 4.3, since the baseline classifier uses an RBF kernel.

- **Asym**: We include the vector difference and vector difference between the LHS and RHS head words, like the model explored in Chapter 3. Note that since the classifier uses an RBF kernel, this is slightly different than our original Asym classifier. However, we tested both and found the difference was marginal.

- **H-features**: We follow the H-feature extraction procedure described in Chapter 4 and extract several iterations of H-features for the LHS and RHS head

words. This is directly comparable to the H-feature model.

Ideally, we should see that the inclusion of these models should provide additional, useful information for the lexical entailment classifier, similar to our findings in the previous chapters. Note that we do not try additional Distributional classifiers, like the Ksim model of Levy et al. (2015b), due to the requirement that the models integrate cleanly with the baseline features. We also note these features are *not* combined with the alignment procedure (Section 6.4.2), as this would increase the number of features to be much larger than the number of labeled examples.

Since we are the first to use these supervised distributional features in any sort of lexical entailment classifier for RTE, all three should be considered original contributions.

6.4.5 Context Vector Features

Finally, we also include *context* similarity features, inspired by the addCos (Melamud et al., 2015b) and nPIC (Roller and Erk, 2016a) models of Lexical Substitution discussed in Chapter 5. These context similarity features required modification of the lexical substitution models, as naive application of either did not produce positive results. Additionally, since we are the first to include context similarity features into the RTE task, these features are an additional contribution of this thesis.

For each pair of head words on the LHS and RHS, we extract all syntactic contexts from the original pair of sentences, and look up their respective *context* vectors from the Context matrix learned by the syntactic distributional space. We then measure five similarities arising from the possible combinations contexts and words: (1) context of LHS with RHS head, (2) context of RHS with LHS head, (3) context of LHS with LHS head, (4) context of RHS with RHS head, and (5) context of LHS with context of RHS.

The first feature is the cosine similarity between the syntactic contexts of the LHS agree with the RHS. This measures how well the RHS's selection preferences agree with the LHS. The reverse corresponds to our second similarity. Based

on the findings of Chapter 2.5, these features correspond roughly to how well the RHS serves as a substitute in the LHS, and vice versa. These features are mostly directly analogous to the addCos model of Melamud et al. (2015b); we also tried *unnormalized* inner products similar to the PIC model, but we found simple cosine performed better.

The third and fourth similarities (LHS contexts with LHS head, RHS contexts with RHS head) do not have an analogous relationship to lexical substitution. However, their purpose is quite intuitive: if the LHS does not strongly agree with its context, then neither should any of its lexical substitutes. In this way, these agreement features do not inform the lexical entailment classifier, except in setting a level of importance for the lexical substitution features.

The final, fifth feature is the similarity between the contexts from the LHS and the contexts of the RHS. This has no relationship to the lexical substitution models, but we found it was important for strong, robust results. This feature works by ensuring that the contexts of the LHS and RHS are similar, and most readily captures tricky RTE phenomenon like subject-object reversal. For example, this feature will identify that the role of 'man' in 'dog bites man' differs substantially from its role in 'man bites dog'. This sort of argument-reversal is one important RTE phenomena which is traditionally ignored in simple alignment and distributional models.

6.5 Experiments and Results

We evaluate all models in two experimental settings, both derived from the Sentences Involving Compositional Knowledge (SICK) dataset, which contains approximately 5000 training and 5000 testing sentence pairs. In the first setting, we evaluate *only* the lexical entailment classifier on its ability to correctly identify lexical pairs as entailing, contradicting or neutral. In this setting, we evaluate only on the lexical entailment rules extracted using the Unification procedure described in Section 6.2. Since only the lexical rules from the training set are annotated, this evaluation is uses a 10-fold cross validation setting. No validation set is used, but

hyperparameters were tuned by maximizing performance across all 10 folds; however, since we are only evaluating on the training set, it is fair to optimize hyperparameters in this way. This experimental setup does not involve the RTE system in any way, and performance is measured in accuracy.

In the second experimental setup, the lexical entailment classifier is trained using all rules extracted from the training set, and then applied to all lexical pairs extracted from the testing set; since the lexical rule extraction procedure is unsupervised, this is the proper way to test generalization to the RTE test set. These test-set lexical predictions are then provided to the RTE system, and the RTE system makes final predictions for the RTE test set. These predictions are evaluated on the RTE dataset directly, so they test how well each of the lexical entailment models contribute as information sources in the logical RTE system. Though the original system described in Beltagy et al. (2016) uses several resources of world knowledge and additional heuristics, in these experiments the lexical entailment classifiers are the *only* source of knowledge. These results are also measured in accuracy.

We emphasize that we do *not* test any of these models in a zero lexical overlap setting. In the experiments of previous chapters, our goal was to test generalization to new lexical items; in these experiments, our goal is to generalize to new RTE examples. Therefore, all of the *sentence* pairs will be unique, but will involve many non-unique lexical entailments: for example, the lexical item 'boy $\rightarrow$ child' appears frequently in both training and test, as both 'boy' and 'child' appear across many sentential pairs.

6.5.1 Individual Components

We first compare each of the individual components of the lexical entailment classifier as how they perform alone, as the single source of lexical knowledge. We also consider one lower and one upper baseline model: the lower baseline model is a majority baseline which always guesses "neutral," so its performance on the RTE Test set can be considered the performance of the *logical system alone*. The upper baseline is given the gold labels for every example, and so its Lexical score is a perfect 100%, but its RTE test score is not; this represents the most any lexical clas-

Model	Lexical	RTE
Neutral (lower baseline)	.643	.735
Gold (upper baseline)	1.000	.968
Simple alignments	.643	.735
Wordform features	.676	.766
Distributional (no binning)	.678	.758
WordNet	.754	.815

(a) Baseline features

Model	Lexical	RTE
Distributional (w/ binning)	.714	.770
Concat (RBF)	.716	.788
Asym	.703	.777
H-features	.716	.784
Context features	.691	.764

(b) Original Contributions

Table 6.3: Comparison of individual lexical entailment components on the Lexical and RTE tasks.

sifier alone can contribute, and any remaining performance is due to inconsistent labeling, errors in logical conclusions, incorrect parsing, or any other imperfections in the end-to-end RTE system. Additionally, we include each of the four sets of features obtained from the work of Lai and Hockenmaier (2014), as comparison points. Finally, we compare each of the our original contribution modules.

The results of all these models may be seen in Table 6.3. We first observe that the Gold lexical baseline has a nonperfect score on RTE test, marking the ceiling of contributions from any of our models, and that it is considerably higher than the lower baseline. This is consistent with the observations of (Dagan et al., 2006), who found Lexical Entailment to be an important component of any RTE system.

We also observe that the strongest individual component, by a large margin, is the WordNet features. This is not surprising, as WordNet is a large, comprehensive and high-quality resource. Additionally, most of the words in SICK contain

WordNet entries, and many of SICK's entries specifically cover simple hypernymy.

We also observe that the Supervised Distributional features substantially outperform the baseline features, with the exception of WordNet. This is consistent with our findings in previous chapters: simple cosine similarity alone is not enough to detect entailment (Weeds et al., 2004; Baroni et al., 2012; Lenci and Benotto, 2012), and there is great deal of benefit using the full distributional vectors in an entailment classifier (Roller et al., 2014; Kruszewski et al., 2015; Roller and Erk, 2016b; Shwartz et al., 2016). Somewhat surprisingly, we find that the Concat and H-features models both perform the same in the Lexical evaluation, but Concat slightly outperforms H-features in the end-to-end RTE evaluation. This emphasizes one important difficulty in using Lexical classifiers in RTE evaluation: many sentence pairs require observing *several* correct lexical entailments to reach the correct, final conclusion. Therefore, small improvements in the lexical entailment classifier may not translate to improvements in the RTE dataset.

We also see that adding the binning to the distributional features substantially improves both the Lexical evaluation and RTE evaluation, compared to the baseline distributional features without binning. In fact, this simple technique brings the distributional similarity classifier to nearly the quality as some of our supervised distributional models. This conforms to our observation in Figure 6.1 and emphasizes that a larger context is important , which found that the conditional probability for entailment differs across different similarity levels.

Finally, we see that the Context features based on Lexical Substitution also outperform many of the individual baseline features, including the distributional features without binning. This emphasizes that integrating a wider context is important in lexical entailment, and that lexical entailment should not be considered in a vacuum.

In the next section, we will combine each of our contributions and consider whether our models may improve upon the Baseline features of prior work, as well as the strengths and weaknesses of our components.

Model	Lexical	RTE
All Baseline Features	.774	.827
+ Binning	.783	.829
+ Contexts	.802	.837
+ Concat	.804	.838
+ Asym	.801	**.844**
+ H-features	**.818**	.834

Table 6.4: SICK performance after adding in our contributions on top of the baseline classifier.

6.5.2 Combining Components

Each of our contributing components discussed above are roughly orthogonal in purpose: binning should capture different entailment likelihoods for different distributional similarities; Context features should capture polysemy or changes in larger context; and Supervised Distributional features should improve generalization to new lexical pairs. Each component should combine to improve overall classification score. Additionally, the Baseline features of prior work are known to already have high performance, so each of our contributions should give improvements over the prior work.

We evaluate combinations of our contributions using a simple ablation experimental setup: first we use the concatenation of all the Baseline features of prior work to find a unified Baseline model provided known good features. Next, we extend the feature set of the baseline model by adding each component one-by-one: first binning, then Contexts, and finally Supervised Distributional models. For the Supervised Distributional models, we consider Concat, Asym, and H-features separately, since they constitute different models. We evaluate each of the experimental conditions using Lexical and RTE accuracies with the same setup as the previous section.

Table 6.4 shows the results of our ablation experiment. We first observe the performance of the Baseline features, which is considerably higher than any of the individual components reported in Table 6.3, emphasizing that we have chosen a

strong Baseline.

Next we consider the addition of Distributional binning on top of the baseline model. We see that Binning provides a strong improvement in lexical classification, and a modest improvement in the RTE evaluation. An analysis of its improvements over the vanilla Baselines model shows that it correctly eliminates false positive classification for some co-hyponym pairs, like 'bed $\nrightarrow$ couch' and 'lawn $\nrightarrow$ field'.

Next we consider how adding the Context features improves over the model with Binning. We again see a substantial improvement in the lexical evaluation, with a more pronounced improvement in the RTE evaluation. We find that the Context features help distinguish cases where altered modifiers (like adjectives or adverbs) make a pair non-entailing. For example, the Context features identifies 'desert area $\nrightarrow$ wooded area', and 'adding $\nrightarrow$ adding slowly'. Unfortunately, this same behavior causes the Contexts model to also misclassify some positive examples where syntactic construction changes radically, like 'wooden hut $\rightarrow$ hut made of wood' and 'making music with flute $\rightarrow$ playing flute'.

Finally, we consider the Supervised Distributional features compared to the model with Context features. Here, we see a small breakdown in the overall pattern: the Asym model actually has a modest *decrease* in performance in the Lexical evaluation but the highest RTE score, and the H-features have a substantial *increase* in lexical evaluation, but an actual decrease in RTE evaluation. Interestingly, an analysis of the results shows that the H-features seem to overwhelm some simpler wordform features: for example, the H-features model incorrectly predicts that 'egg $\rightarrow$ two eggs', indicating the model has "forgotten" how to handle plurality. In other cases, the H-features model makes an arguably correct lexical entailment, with a resulting incorrect RTE evaluation: for example, the H-feature model and Asym model differ in predictions about 'young woman $\rightarrow$ girl', with the H-feature model believing this is a nonentailment. We suspect different annotators may have different opinions about which is correct.

We also note there are a handful of systematic lexical entailment differences between the training and test set, which may account for some of the disconnect

in performance: for example, the training set frequently contains the annotation 'woman $\rightarrow$ man', while the test set contains only 'woman $\nrightarrow$ man'. If a model learns to classify this pair as an entailment, then the lexical evaluation will increase, since the lexical evaluation is done only on the training set. However, RTE evaluation will also decrease, since it is evaluated using the Test set.

Nonetheless, despite these complications, we do generally see improvements from the addition of Supervised Distributional features. Furthermore, all of our Contribution models improve over the Baseline classifier in both Lexical and RTE evaluations, corroborating our findings in the other chapters of this thesis.

6.6 Chapter Summary

In this chapter, we considered the role of lexical entailment classifiers in an end-to-end RTE task, and proposed three directions where improvements in lexical entailment could lead to improvements in sentential entailment.

Our first contribution is distributional binning, which improves upon the false positive rate of models which employ simple distributional similarity. We observe that the probability of entailment is non-monotonic with respect to the cosine similarity of a word pair. That is, more similar words are more likely to be entailing, except the most highly similar words tend more often to be co-hyponyms, and therefore nonentailing. By grouping similarities into discrete levels, we can capture this phenomenon and improve performance.

Our second contribution comes from integrating in the wider context of a lexical entailment through the use of Context vectors. We use a similar procedure as the one discussed in our Lexical Substitution chapter, where the syntactic contexts of a target word are additionally extracted and represented using their corresponding entries from the distributional context matrix. By adding context similarities into our model, we correctly identify some nonentailments derived from the addition of modifiers, but also make some misclassifications when syntactic constructions differ considerably.

Finally, we integrate in the Concat, Asym and H-feature models discussed

in previous chapters. We find that the integration of H-features into the model substantially improves performance at the lexical level, but results in slightly lower performance at the full RTE level. Systematic differences between the training and test set may account for some of this discrepancy. Furthermore, we find that the addition of the Asym features substantially improves RTE accuracy, and gives the strongest performance of any of our considered models. In general, we find that Supervised Distributional models can contribute to an end-to-end RTE system.

Chapter 7
Conclusion

Distributional Semantics has come a long way in its ability to contribute to difficult Natural Language Processing tasks. Distributional representations of word meaning have been successfully used on a wide variety of lexical semantics tasks, and have become the shoulders on which modern NLP methods stand (Goldberg, 2016).

In this dissertation, we considered how distributional representations of word meaning can be useful for identifying and exploiting *lexical entailment*. We have considered a variety of challenging tasks related to the area, including hypernymy detection, lexical relationship prediction, and lexical substitution.

In hypernymy detection and lexical relationship prediction, we predict whether a given pair of words exhibits a particular linguistic relationship, such as hypernymy, co-hyponymy, or meronymy. Our work has shown that the choice of experimental setup is critical to properly understanding how methods may or may not generalize to novel lexical items, and introduces the notion of lexical memorization. We proposed evaluating models in an adversarial setup with zero lexical overlap between training and test sets, allowing us to measure generalization to unseen words, and our setup has become a standard evaluation in the literature.

We also proposed Asym, a new model of hypernymy detection and lexical relationship prediction. We analyzed Asym to identify its relationship to existing linguistic theories of hypernymy, like the Distributional Inclusion Hypothesis. We showed that our model does not suffer from some of the prototypicality limitations of other models proposed in the literature, which only make predictions based on how similar a pair is to relationship prototypes, without regard for actual relationship between the words in a pair.

We considered the behavior of a model known to exhibit *only* prototypicality behavior, and proposed a novel analysis method to interpret the behavior of the model. By interpreting a model's hyperplane in terms of *context* space, we ob-

served that simpler models actually learn to identify H-features, or Hearst pattern like contexts which are most indicative of hypernymy. Building on this observation and the success of Asym, we proposed a novel model which identifies and exploits multiple sets of H-features through an iterative PCA-like procedure. Our model matches or exceeds the performance of other models in the literature on several datasets. We also extended our H-feature model so that it may predict several lexical relationships simultaneously and outperform existing models in the literature. We examined the H-features learned for the non-hypernymy relationships and found strong evidence for additional known Hearst patterns in the literature, as well as many alternative patterns highly indicative of target relationships.

We considered a novel model of lexical substitution, the task of predicting a paraphrase for a polysemous word in a particular sentential context. We introduced a modification to a simple model from he literature, and showed large performance gains over several datasets and evaluations compared to the baseline model. Our performance most improved on the difficult task of substitution generation, where a one may propose any word from the entire vocabulary as a substitute. Additional analysis showed that our model improved over prior work via integration of a unigram frequency bias, allowing it to discard rare or misspelled substitutes from its predictions.

Finally, we considered how each of the contributions above could be integrated into a real, end-to-end system for Recognizing Textual Entailment. We considered the role of a lexical entailment classifier in an RTE system, and the relationship between distributional similarity and the probability of lexical entailment. We found that each of our components, on its own, was able to perform competitively or outperform a number of baseline lexical entailment features. Furthermore, we showed that the components may be combined with existing lexical entailment features to improve overall performance, significantly outperforming a system which used only a fixed, high-quality lexical resource. Our contributing components also combine together to produce a model which outperforms any component on its own.

In short, this thesis has shown that Distributional Semantics can contribute significantly to difficult lexical and textual entailment tasks through a variety of

techniques and models, and that efforts on each task have yielded a deeper understanding about distributional word representations and the information contained within them.

7.1 Future Work

As with all areas of research, the work in this thesis answers some questions, but it also raises new ones. In this section, we briefly consider some possible future directions of research.

Long-distance Dependency Contexts

Throughout this thesis, we saw that a variety of syntactic contexts can play important roles in the predictive power of models. For example, in the work on H-features, we saw that some complex syntactic contexts, like `nmod:between`$^{-1}$`+cross` was one context learned to be indicative of co-hyponymy, and `nmod:from`$^{-1}$`+handcraft` was learned to be indicative of constructive material. These contexts were extracted due to the collapsed dependency structures used at the time of space creation (de Marneffe and Manning, 2008). The dependencies that are collapsed are based on a small list of fixed, English multiword expressions. As such, other, more complex syntactic fragments, or fragments with multiple intermediary points may form additional strong signal for the applications explored in this thesis. Models that use *paths* through dependency parses have had success in lexical relationship prediction (Shwartz et al., 2016; Shwartz and Dagan, 2016a) and semantic role labeling (Roth and Lapata, 2016). Models of these paths are likely useful as proxies for wider-context information in a future distributional models.

Multi-relation Lexicalization

In a similar vein, current work on distributional models treats each co-occurrence as an independent, isolated event in the corpus. For example, in our distributional spaces, we model verb-subject co-occurrences independently of verb-

object co-occurrences independently of verb-preposition co-occurrences. In some cases, it may be better to model a word's co-occurrences jointly. For example, if someone "kills two birds," we may have an industrious hunter, and if we observe someone "killing with one stone," we may recall the story of David and Goliath. However, if we observe someone "killing two birds with one stone," the situation is entirely different from the previous two scenarios. It follows that some words or specific co-occurrences may be better modeled if two co-occurring *contexts* are also modeled as a single unit, rather than separate, smaller contexts. Indeed, Chersoni et al. (2016) find that jointly modeling a verb using its subject *and* object gives better estimates of human similarity scores, but we suspect joint modeling may be valuable for several combinations of syntactic relationships.

Sparsity will always be a fundamental challenge in such joint-modeling applications, but clustering methods alleviate this problem by using coarser co-occurrences for modeling. Melamud et al. (2014) makes a step in this direction by using probability estimates of a language model to estimate joint co-occurrence, but they ignore explicit markers of syntactic relations. Clustering approaches may also be helpful if used as substitutes for words occurring in particular relations, giving coarser groups of co-occurrences. However, they may result in the loss of idiomatic constructions, like our example, or cause overgeneralization from idioms, such as "murdering two animals with one rock."

Joint H-feature Learning

In Chapter 4, we used an iterative, progressive procedure for extracting multiple H-features, with the second H-feature only being extracted after the first is individually modeled and subtracted from the data. However, this greedy procedure possibly results in capturing less-than-ideal H-features. In our own examples, we saw that the H-features related to vehicles were mixed with H-features of common tools, though finding two separate groups of H-features is likely better. One solution to this problem would be to learn to extract all H-features *simultaneously* and *jointly*, rather than using our iterative greedy procedure. This could be done

using a neural network model, such that the first layer's weight matrix correspond to each separate H-feature. In preliminary experiments testing this approach, we struggled with overfitting on specific lexical items, and suffered a great deal of the same lexical memorization issue of early models in the area. It may be necessary to regularize the model using orthogonal regularization (Brock et al., 2015), or a sparsity regularizer on hidden activations.

Lexical Relationship Generation

In the broader context, predicting whether two words exhibit a lexical relationship is of limited use when one or both sides of the pair is unconstrained, and the best possible word must be *generated* or *selected* from the vocabulary. For example, some applications may be actually more interested in asking "what is the hypernym of 'cat'?" rather than "is ('cat', 'animal') hypernymy?" At the present moment, our H-feature model and similar lexical relationship models could only answer the former question by enumerating over the vocabulary, and querying the model word by word. Naturally, even if we improved our predictive accuracy substantially beyond current state-of-the-art, the the numerous trials will result in a large number of false positives.

Furthermore, we saw in our own experiments that highly imbalanced datasets (like the Medical dataset) have much lower accuracies than more balanced datasets, indicating the problem cannot be solved via data augmentation alone. As such, distributional lexical relationship *generation* is an interesting area of further research with limited prior work (Fu et al., 2014; Nayak, 2015; Espinosa Anke et al., 2016). We hope future researchers in this area will benefit from the lessons learned in this thesis.

Structured Relationship Prediction

In the same vein as Lexical Relationship Generation, we should also recall that many interesting linguistic relationships are actually interconnected, structured problems. For example, co-hyponymy is a useful linguistic relationship with its own

interesting signals and Hearst patterns (as we saw in Chapter 4), but it is also inseparably defined in terms of hypernymy. Yet, the models discussed in this thesis do not consider relationship prediction in this manner: (‘cat’, ‘animal’) is classified independently of (‘dog’, ‘animal’) and (‘cat’, ‘dog’). A good system of lexical relationship prediction, or generation, should weigh the evidence for all three pairs together, and come to a conclusion as a whole. That is, models should also be forced to consider the natural constraints of the taxonomic properties, in addition to evidence between two individual pairs. Snow et al. (2006) showed that evidence from different relations can be weighted and combined to produce correctly structured taxonomies, but an ideal work would bake the structure into the original model.

Soft Alignments in Lexical Entailment

In our Lexical Entailment classifier of Chapter 6, we used a hard alignment procedure in order to model entailment aspects of the non-head words in phrases Lai and Hockenmaier (2014). Although this successfully models some entailing and nonentailing phrases, the greedy alignment procedure may sometimes cause words to wrongly become aligned, and prevents any many-to-one alignments. Future models may benefit from using a soft alignment model, similar to the attention mechanisms of neural methods in RTE (Bowman et al., 2015; Parikh et al., 2016). This could be accomplished by integrating soft alignments into our lexical entailment classifier, or by integrating our features into attention-based neural networks for RTE.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. A study on similarity and relatedness using distributional and WordNet-based approaches. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, Boulder, CO, 2009.

Ali Mohamed Nabil Allam and Mohamed Hassan Haggag. The question answering systems: A survey. *International Journal of Research and Reviews in Information Sciences*, 2(3), 2012.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 86–90. Association for Computational Linguistics, 1998.

Colin Bannard and Chris Callison-Burch. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 597–604, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

Marco Baroni and Alessandro Lenci. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721, 2010.

Marco Baroni and Alessandro Lenci. How we BLESSed distributional semantic evaluation. In *Proceedings of the 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10, Edinburgh, UK, 2011.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. Entailment above the word level in distributional semantics. In *Proceedings of the 2012 Conference of the European Chapter of the Association for Computational Linguists*, pages 23–32, Avignon, France, 2012.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 238–247, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

I. Beltagy, Stephen Roller, Gemma Boleda, Katrin Erk, and Raymond J. Mooney. Utexas: Natural language semantics using distributional semantics and probabilistic logic. In *Proceedings of the Eighth International Workshop on Semantic Evaluation*, pages 796–801, Dublin, Ireland, August 2014. Association for Computational Linguistics and Dublin City University.

I. Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. Representing meaning with a combination of logical and distributional models. *Special Issue of Computational Linguistics on Formal Distributional Semantics*, 2016.

I. Beltagy. *Natural Language Semantics Using Probabilistic Logic*. PhD thesis, Department of Computer Science, The University of Texas at Austin, December 2016.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. The fifth PASCAL Recognizing Textual Entailment challenge. *Proceedings of the Text Analytics Conference*, 9:14–24, 2009.

Chris Biemann. Turk bootstrap word sense inventory 2.0: A large-scale resource for lexical substitution. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, pages 4038–4042, Istanbul, Turkey, May 2012. European Language Resources Association.

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python*. O'Reilly Media, Inc., 2009.

Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the Eighth International Workshop on Semantic Evaluation*, pages 642–646, Dublin, Ireland, August 2014. Association for Computational Linguistics and Dublin City University.

David Blei, Andrew Y. Ng, and Michael Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

Johan Bos. Wide-coverage semantic analysis with Boxer. In *Proceedings of Semantics in Text Processing*, 2008.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings*

of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1466–1477, Berlin, Germany, August 2016. Association for Computational Linguistics.

Andrew Brock, Theodore Lim, J. M. Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. In *Proceedings of 2017 International Conference on Learning Representations*, 2015.

John A. Bullinaria and Joseph P. Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*, 39(3):510–526, 2007.

Danqi Chen and Christopher D. Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 740–750, Doha, Qatar, October 2014. Association for Computational Linguistics.

Jianpeng Cheng, Li Dong, and Mirella Lapata. Long Short-Term Memory-Networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas, November 2016. Association for Computational Linguistics.

Emmanuele Chersoni, Enrico Santus, Alessandro Lenci, Philippe Blache, and Chu-Ren Huang. Representing verbs with rich contexts: An evaluation on verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1967–1972, Austin, Texas, November 2016. Association for Computational Linguistics.

Kyunghyun Cho. Natural language understanding with distributed representation. *arXiv preprint arXiv:1511.07916*, 2015.

Daoud Clarke. Context-theoretic semantics for natural language: An overview. In *Proceedings of the 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 112–119, Athens, Greece, March 2009. Association for Computational Linguistics.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Ido Dagan, Oren Glickman, and Bernardo Magnini. *The PASCAL Recognising Textual Entailment Challenge*, pages 177–190. Springer, Berlin, Heidelberg, 2006.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. Recognizing Textual Entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220, 2013.

Marie-Catherine de Marneffe and Christopher D. Manning. Stanford typed dependencies manual. Technical report, Stanford University, 2008.

Scott Deerwester, Susan Dumais, Thomas Landauer, George Furnas, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–407, 1990.

Mona Diab. *Word sense disambiguation within a multilingual framework*. PhD thesis, University of Maryland, 2003.

Georgiana Dinu and Mirella Lapata. Measuring distributional similarity in context. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1162–1172, Cambridge, MA, October 2010. Association for Computational Linguistics.

Katrin Erk and Sebastian Padó. A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 897–906, Honolulu, Hawaii, October 2008. Association for Computational Linguistics.

Katrin Erk. What is word meaning, really? (and how can distributional models help us describe it?). In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, pages 17–26, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

Katrin Erk. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653, 2012.

Luis Espinosa Anke, Jose Camacho-Collados, Claudio Delli Bovi, and Horacio Saggion. Supervised distributional hypernym discovery via domain adaptation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 424–435, Austin, Texas, November 2016. Association for Computational Linguistics.

Stefan Evert. *The statistics of word cooccurrences: word pairs and collocations*. PhD thesis, Stuttgart University, 2005.

John R. Firth. A synopsis of linguistic theory 1930–1955. In *Studies in linguistic analysis*, pages 1–32. Blackwell Publishers, Oxford, England, 1957.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. Learning semantic hierarchies via word embeddings. In *Proceedings of the 2014 Annual Meeting of the Association for Computational Linguistics*, pages 1199–1209, Baltimore, Maryland, 2014.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The paraphrase database. In *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013.

Maayan Geffet and Ido Dagan. Feature vector quality and distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 247. Association for Computational Linguistics, 2004.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL Recognizing Textual Entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics, 2007.

Roxana Girju, Adriana Badulescu, and Dan Moldovan. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1):83–135, March 2006.

Claudio Giuliano, Alfio Gliozzo, and Carlo Strapparava. FBK-irst: Lexical substitution task exploiting domain and syntagmatic coherence. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*, pages 145–148, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

Poonam Gupta and Vishal Gupta. A survey of text question answering techniques. *International Journal of Computer Applications*, 53(4), 2012.

Zellig S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 1992 Conference on Computational Linguistics*, pages 539–545, Nantes, France, 1992.

Lynette Hirschman and Robert Gaizauskas. Natural language question answering: the view from here. *Natural Language Engineering*, 7(04):275–300, 2001.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.

Eric Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 873–882, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the Seventh International Joint Conference on Natural Language Processing*, pages 1–10, Beijing, China, July 2015. Association for Computational Linguistics.

David A. Jurgens, Peter D. Turney, Saif M. Mohammad, and Keith J. Holyoak. SemEval-2012 Task 2: Measuring degrees of relational similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 356–364. Association for Computational Linguistics, 2012.

Kazuya Kawakami and Chris Dyer. Learning to represent words in context with multilingual supervision. In *Proceedings of 2016 International Conference on Learning Representations*, 2016.

Adam Kilgarriff and Joseph Rosenzweig. Framework and results for english senseval. *Computers and the Humanities*, 34(1):15–48, 2000.

Kazuaki Kishida. Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments. *NII Technical Reports*, 2005(14):1–19, 9 2005.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16:359–389, 10 2010.

Zornitsa Kozareva and Eduard Hovy. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1110–1118, Cambridge, MA, October 2010. Association for Computational Linguistics.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. What substitutes tell us - analysis of an "all-words" lexical substitution corpus. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 540–549, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

Germán Kruszewski, Denis Paperno, and Marco Baroni. Deriving Boolean structures from distributional vectors. *Transactions of the Association for Computational Linguistics*, 3:375–388, 2015.

Alice Lai and Julia Hockenmaier. Illinois-LH: A denotational and distributional approach to semantics. In *Proceedings of the Eighth International Workshop on Semantic Evaluation*, pages 329–334, Dublin, Ireland, August 2014. Association for Computational Linguistics and Dublin City University.

Thomas K. Landauer and Susan T. Dumais. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211, 1997.

Alessandro Lenci and Giulia Benotto. Identifying hypernyms in distributional semantic spaces. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 75–79, Montréal, Canada, June 2012. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 302–308, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc., 2014.

Omer Levy, Ido Dagan, and Jacob Goldberger. Focused entailment graphs for Open IE propositions. In *Proceedings of the 2014 Conference on Computational Natural Language Learning*, pages 87–97, Ann Arbor, Michigan, 2014.

Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the 2014 Conference on Computational Natural Language Learning*, pages 171–180, 2014.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, Denver, Colorado, 2015.

Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the 14th International Conference on Machine Learning*, volume 98, pages 296–304, 1998.

Kevin Lund and Curt Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208, 1996.

Bill MacCartney and Christopher D. Manning. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 521–528. Association for Computational Linguistics, 2008.

Christopher D. Manning. Computational linguistics and deep learning. *Computational Linguistics*, 41:701–707, 2015.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. SemEval-2014 Task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the Eighth International Workshop on Semantic Evaluation*, pages 1–8, Dublin, Ireland, 2014.

Diana McCarthy and Roberto Navigli. SemEval-2007 Task 10: English lexical substitution task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*, pages 48–53, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Diana McCarthy. Word sense disambiguation: An overview. *Language and Linguistics compass*, 3(2):537–558, 2009.

Oren Melamud, Ido Dagan, Jacob Goldberger, Idan Szpektor, and Deniz Yuret. Probabilistic modeling of joint-context in distributional similarity. In *Proceedings of the 18th Conference on Computational Natural Language Learning*, pages 181–190, Ann Arbor, Michigan, June 2014. Association for Computational Linguistics.

Oren Melamud, Ido Dagan, and Jacob Goldberger. Modeling word meaning in context with substitute vectors. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 472–482, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

Oren Melamud, Omer Levy, and Ido Dagan. A simple word embedding model for lexical substitution. In *Proceedings of the First Workshop on Vector Space Modeling for Natural Language Processing*, pages 1–7, Denver, Colorado, June 2015. Association for Computational Linguistics.

Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany, August 2016. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of 2013 International Conference on Learning Representations*, 2013.

George A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 130–136, Berlin, Germany, August 2016. Association for Computational Linguistics.

Roberto Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10, 2009.

Neha Nayak. Learning hypernymy over word embeddings. Technical report, Stanford, 2015.

Silvia Necsulescu, Sara Mendes, David Jurgens, Núria Bel, and Roberto Navigli. Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 182–192, Denver, Colorado, June 2015. Association for Computational Linguistics.

Hwee Tou Ng, Chung Yong Lim, and Shou King Foo. A case study on inter-annotator agreement for word sense disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Standardizing Lexical Resources*, College Park, Maryland, 1999.

Sebastian Padó and Mirella Lapata. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199, 2007.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas, November 2016. Association for Computational Linguistics.

Ellie Pavlick and Chris Callison-Burch. Most "babies" are "little" and most "problems" are "huge": Compositional entailment in adjective-nouns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2164–2173, Berlin, Germany, August 2016. Association for Computational Linguistics.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Yves Peirsman. Word space models of semantic similarity and relatedness. In *European Summer School in Logic, Language and Information Student Session*, Hamburg, Germany, 2008.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

Joseph Reisinger and Raymond J. Mooney. Multi-prototype vector-space models of word meaning. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 109–117. Association for Computational Linguistics, 2010.

Philip Resnik and David Yarowsky. Distinguishing systems and distinguishing senses: New evaluation methods for word sense disambiguation. *Natural language engineering*, 5(02):113–133, 1999.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

Alan Ritter, Stephen Soderland, and Oren Etzioni. What is this, anyway: Automatic hypernym discovery. In *AAAI Spring Symposium: Learning by Reading and Learning to Read*, pages 88–93, 2009.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, and Phil Blunsom. Reasoning about entailment with neural attention. In *Proceedings of 2016 International Conference on Learning Representations*, 2016.

Stephen Roller and Katrin Erk. PIC a different word: A simple model for lexical substitution in context. In *Proceedings of the 2016 North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2016.

Stephen Roller and Katrin Erk. Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, USA, November 2016. Association for Computational Linguistics.

Stephen Roller, Katrin Erk, and Gemma Boleda. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the 2014 International Conference on Computational Linguistics*, pages 1025–1036, Dublin, Ireland, 2014.

Michael Roth and Mirella Lapata. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1192–1202, Berlin, Germany, August 2016. Association for Computational Linguistics.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 38–42, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. EVALution 1.0: An evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of the Fourth Workshop on Linked Data in Linguistics: Resources and Applications*, pages 64–69, Beijing, China, July 2015. Association for Computational Linguistics.

Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. Nine features in a random forest to learn taxonomical semantic relations. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 10th International Conference on Language Resources and Evaluation*, Paris, France, May 2016. European Language Resources Association.

Enrico Santus. SLQS: An entropy measure. Master's thesis, University of Pisa, 2013.

Julian Seitner, Christian Bizer, Kai Eckert, Stefano Faralli, Robert Meusel, Heiko Paulheim, and Simone Ponzetto. A large database of hypernymy relations extracted from the web. In *Proceedings of the 10th edition of the Language Resources and Evaluation Conference, Portoroz, Slovenia*, 2016.

Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948.

Eyal Shnarch. Lexical entailment and its extraction from wikipedia. Master's thesis, Bar-Ilan University, 2008.

Vered Shwartz and Ido Dagan. CogALex-V Shared Task: LexNET - integrated path-based and distributional method for the identification of semantic relations. In *Proceedings of the Fifth Workshop on Cognitive Aspects of the Lexicon (CogALex-V)*, pages 80–85, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

Vered Shwartz and Ido Dagan. Path-based vs. distributional information in recognizing lexical semantic relations. In *Proceedings of the Fifth Workshop on Cognitive Aspects of the Lexicon (CogALex-V)*, pages 24–29, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2389–2398, Berlin, Germany, August 2016. Association for Computational Linguistics.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 65–75, Valencia, Spain, April 2017. Association for Computational Linguistics.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1297–1304. MIT Press, 2004.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 801–808. Association for Computational Linguistics, 2006.

György Szarvas, Chris Biemann, and Iryna Gurevych. Supervised all-words lexical substitution using delexicalized features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1131–1141, 2013.

György Szarvas, Róbert Busa-Fekete, and Eyke Hüllermeier. Learning to rank lexical substitutions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1926–1932, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

Stefan Thater, Georgiana Dinu, and Manfred Pinkal. Ranking paraphrases in context. In *Proceedings of the 2009 Workshop on Applied Textual Inference*, pages

44–47, Suntec, Singapore, August 2009. Association for Computational Linguistics.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 948–957, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. Word meaning in context: A simple and effective vector model. In *Proceedings of Fifth International Joint Conference on Natural Language Processing*, pages 1134–1143, Chiang Mai, Thailand, November 2011. Asian Federation of Natural Language Processing.

Erik Tjong Kim Sang. Extracting hypernym pairs from the web. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion*, pages 165–168, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Lloyd N. Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

Peter Turney and Saif Mohammad. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering*, 21(03):437–476, 2015.

Peter Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, 2010.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. Latent vector weighting for word meaning in context. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1012–1022, Edinburgh, Scotland, UK, July 2011. Association for Computational Linguistics.

Ivan Vendrov, Jamie Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. In *Proceedings of 2016 International Conference on Learning Representations*, 2016.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1671–1682, Berlin, Germany, August 2016. Association for Computational Linguistics.

Julie Weeds and David Weir. A general framework for distributional similarity. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 81–88, 2003.

Julie Weeds, David Weir, and Diana McCarthy. Characterising measures of lexical distributional similarity. In *Proceedings of the 2004 International Conference on Computational Linguistics*, pages 1015–1021, Geneva, Switzerland, 2004.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of the 2014 International Conference on Computational Linguistics*, pages 2249–2259, Dublin, Ireland, 2014.

Benjamin J. Wilson and Adriaan M. J. Schakel. Controlled experiments for word embeddings. *arXiv preprint arXiv:1510.02675*, 2015.

Ludwig Wittgenstein. *Philosophical investigations*. Blackwell Publishers, Oxford, England, 1953.

Deniz Yuret. KU: Word sense disambiguation by substitution. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*, pages 207–214, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Maayan Zhitomirsky-Geffet and Ido Dagan. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 2005 Annual Meeting of the Association for Computational Linguistics*, pages 107–114, Ann Arbor, Michigan, 2005.

Vita

Stephen Roller was born in the early hours of a winter's day in 1988 in Baltimore, Maryland. He spent his formative years in Charlotte, North Carolina. He obtained a Bachelors of Science in Computer Science from North Carolina State University in 2010, and then entered graduate school at the University of Texas at Austin. He obtained a Masters of Science in Computer Science in 2014, and continued his doctoral studies, where he has been working on natural language understanding.

Permanent Address: me@stephenroller.com

This dissertation was typeset by the author with LaTeX.