PARAPHRASE-BASED MODELS OF LEXICAL SEMANTICS

Anne O'Donnell Cocos

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2019

For my grandmothers, Barbara and Ora,

who were both women ahead of their time.

ACKNOWLEDGEMENT

I am not sure that there are many faculty who would take a chance on a Ph.D. applicant who is eager, but comes from a very unconventional background with significant gaps in her formal computer science education. I am grateful to my advisor, Chris Callison-Burch, for being one of them. Chris has been incredibly supportive from day one, providing just the right amount of direction or leeway as needed. Most of all, I admire and appreciate him for actively fostering such a welcoming, diverse, and collaborative atmosphere in the lab – one that I hope to emulate in my own career. It has been a joy to work together these last five years.

I am also indebted to the members of my thesis committee – Marianna Apidianaki, Mitch Marcus, Dan Roth, and Lyle Ungar – for shaping this work for the better. Marianna has encouraged me to explore all angles of an idea, helping to validate and solidify thoughts and arguments. Mitch is a fount of wisdom on both research and life in general, and I have been the grateful recipient of both. Dan's enthusiasm for ideas has encouraged me as a new researcher, and the addition of his lab has made the Penn NLP community all the more vibrant. Lyle's direct and constructive questions have been instrumental in helping me to clarify and articulate my thoughts.

I am grateful to the National Physical Sciences Consortium, the Allen Institute for Artificial Intelligence, Student Veterans of America, and the Google Ph.D. Fellowship program for their generous support throughout my time at Penn. A special thanks goes to Dipanjan Das for mentoring me over the last year – our conversations on research and managing research have been very helpful, and I appreciate his genuine care throughout the process of my job search.

One of the most satisfying parts of my work so far has been the opportunity to apply NLP to pediatric medical research with Aaron Masino and his lab at The Children's Hospital of Philadelphia. Aaron is a wonderful supervisor and mentor, and working with clinicians at

CHOP has shaped my approach to thinking about AI in the ‘real world.’ Thank you to all my collaborators there, especially Ting and Daniel, for everything you have taught me.

A huge thanks to the other students, postdocs, and researchers at Penn, past and present, who make me look forward to heading in to 3401 Walnut each day. To Ellie, João, Jessy, Wei, Derry, Andy, Marianna, Reno, Jie, Jordan, Daphne, Maria, Rebecca, Nitish, Shyam, Stephen, Hangfeng, Dan, Daniel, Danni, and Jennifer: I feel very lucky that all of you friendly and brilliant people have crossed my path, and I am excited to watch where your ideas take you. Also, thank you to Mike Felker for looking out for me since the day I applied to Penn, and for making me feel immediately at home within our tiny CIS vets community. And to Sheryl, thank you for being a friendly face each morning and for making the office a comfortable place to work.

Finishing my Ph.D. would simply not have been possible without an amazing support network behind the scenes. There is a team of people who enabled me to focus on work by providing a warm and safe environment for my boys at home, especially Becca, Delaney, Linda, Tanya, and all the wonderful women and men at SPNDS. Then, of course, my friends and family, who are daily reminders of what is important. To my parents, thank you for teaching me to be confident in taking chances, knowing you had my back, and for providing back-up help on countless occasions these last few years. To my in-laws, Barb and Bill, who are so supportive and willing to fly halfway around the world to help us out in a pinch. To Gabe and Jack, thank you for making me smile every day. I am so proud of how curious and kind you are both growing to be. And last, but certainly not least, to Will, my dearest friend, thank you for being the rock of our family and always keeping me grounded. Love to you all.

ABSTRACT

PARAPHRASE-BASED MODELS OF LEXICAL SEMANTICS

Anne O'Donnell Cocos

Chris Callison-Burch

Models of lexical semantics are a key component of natural language understanding. The bulk of work in this area has focused on learning the meanings of words and phrases and their inter-relationships from signals present in large monolingual corpora – including the distributional properties of words and phrases, and the lexico-syntactic patterns within which they appear. Each of these signals, while useful, has drawbacks related to challenges in modeling polysemy or limited coverage. The goal of this thesis is to examine bilingually-induced paraphrases as a different and complementary source of information for building computational models of semantics.

First, focusing on the two tasks of discriminating word sense and predicting scalar adjective intensity, we build models that rely on paraphrases as a source of signal. In each case, the performance of the paraphrase-based models is compared to that of models incorporating more traditional feature types, such as monolingual distributional similarity and lexico-syntactic patterns. We find that combining these traditional signals with paraphrase-based features leads to the highest performing models overall, indicating that the different types of information are complementary. Next, we shift focus to the use of paraphrases to model the fine-grained meanings of a word. This idea is leveraged to automatically generate a large resource of meaning-specific word instances called Paraphrase-Sense-Tagged Sentences (PSTS). Distributional models for sense embedding, word sense induction, and contextual hypernym prediction are trained successfully by using PSTS as a sense-tagged corpus. In this way we reaffirm the notion that signals from paraphrases and monolingual distributional properties can be combined to construct robust models of lexical semantics.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER 1 : Introduction

1.1. Overview

When we hear this question as humans:

> *What is a Chinese dish that's not so hot?*

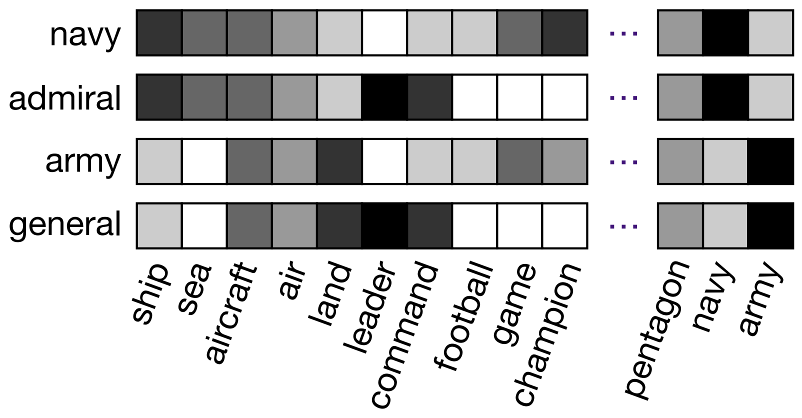
we understand the question in the context of the real world – many types of Chinese food are spicy, and the questioner is looking for a meal that is mild in flavor. An automated question answering (QA) system, however, cannot frame the question in this particular context without an underlying model of semantics. In particular, in order to give a satisfactory answer, the QA system must have some way to deal with polysemy (*hot dish* refers to spicy food, not a stolen satellite dish), hypernymy (*sweet & sour pork* is a kind of *Chinese dish*), and scalar adjective intensity (*zesty* and *peppery* dishes are fine answers; *fiery* ones are not).

Word sense, hypernymy, and scalar adjective intensity are all aspects of *lexical semantics*, which deals with the meanings of and relationships between terms. Lexical semantics are building blocks of natural language understanding. In order for a computer to interpret, reason about, and generate text, it must have a mechanism to model the semantics of individual terms – both their meanings, and inter-relationships.

Attempts to model lexical semantics have involved both manual and automatic methods. With respect to the former, there exist several well known and widely-used hand-compiled ontologies. These include general-purpose resources such as WordNet (Miller, 1995) and EuroWordNet (Vossen, 2004), and domain-specific ontologies like the Unified Medical Language System (UMLS) (Bodenreider, 2004). These ontologies are collections of entities, organized via pairwise relations. One benefit of using hand-crafted ontologies to model lexical semantics is that they have a clean structure with precisely defined relations (e.g. hypernymy, meronymy, etc). Another benefit is that entities are encoded at the word sense level; the noun *dish* maps to six distinct WordNet entities, which capture its satellite receiver and dinnerware senses, among others. The primary drawback to using manually-compiled ontologies to model semantics is that they are expensive to create, making them difficult to update or adapt to new domains. Another disadvantage is their limited coverage. For example, WordNet includes 155k unique terms. Although this may seem high, it represents only a fraction of the English vocabulary; the Google N-grams corpus (Thorsten and Franz, 2006) contains over ten times as many English unigrams and bigrams that occur at least 50k times on the web.

In order to overcome the shortfalls of manually-generated resources, researchers have devised automated methods to learn the meanings of and relationships between terms. Common automatic methods incorporate monolingual signals such as contextual similarity and lexico-syntactic patterns (Figure 1). Models based on contextual similarity are grounded in the intuition that semantically related words tend to appear within similar contexts (Harris, 1954). This single idea has formed the basis for much of the progress in computational lexical semantics to date. But contextual similarity provides only a fuzzy signal of semantic relatedness; pairs of terms with similar contextual representations might be more precisely classified as synonyms, hypernyms, meronyms, or even antonyms, but further analysis is required to determine which specific relation holds. Additionally, word representations that are built upon monolingual context tend to be dominated by the most frequent sense of a word, and may fail to capture more infrequent meanings. Lexico-syntactic patterns, on the other hand, are textual templates that are indicative of a particular semantic relationship, like the pattern *"Y, such as X"* which suggests that X is a hyponym of Y (Hearst, 1992). They can be used to precisely identify term pairs that are hypernyms, meronyms, or adjectives of varying intensity describing a shared attribute. But some relation types, such as synonymy, are not indicated through patterns in text. Additionally, pattern-based methods obfuscate word sense; it is unclear on the surface whether the *great* in *"[good], but not [great]"* refers to quality or size.

This thesis explores the use of a third type of signal – paraphrases – for learning lexical



(a) Word vectors encoding context similarity. Each row represents one word, and each column corresponds to a possible context. Darker cells indicate higher affinity. Semantically similar words can be expected to occur in similar contexts, and therefore have similar vector representations.

Pattern	*Instance*	*Extracted Relation*
`X, such as Y`	Some **[ships], such as [frigates]**, were built for speed.	`frigate IS-A ship` (hypernymy)
`Y, although not X`	The film was **[funny], although not [hilarious]**.	`funny < hilarious` (adjective intensity)

(b) Lexical syntactic patterns mined from text to discover hypernyms (top) and relative adjective intensity (bottom)

Figure 1: Contextual similarity and lexico-syntactic patterns are common signals derived from monolingual corpora that can be used to encode word meaning and discover semantic relationships.

semantics. Paraphrases are differing textual expressions, or surface forms, in the same language with approximately the same meaning (Madnani and Dorr, 2010; Bhagat and Hovy, 2013). They are useful in a number of tasks such as question answering and information retrieval (Navigli and Velardi, 2003; Riezler et al., 2007), evaluating machine translation (Denkowski and Lavie, 2010), and recognizing textual entailment (Pavlick et al., 2015a). In general, paraphrases can be generated at large scale using either monolingual or bilingual methods (Madnani and Dorr, 2010). In this thesis, we focus solely on paraphrases that have been extracted from bilingual parallel corpora using a method called "bilingual pivoting" (Bannard and Callison-Burch, 2005; Callison-Burch, 2008), which is motivated by the idea that two English terms that share multiple foreign translations are likely to have similar meaning.

Definition 1.1.1: Paraphrase

Paraphrases are differing surface forms with approximately the same meaning. In this work we refer specifically to paraphrases derived from *bilingual pivoting*, based upon the premise that two English terms sharing multiple foreign translations are likely to have similar meaning (Bannard and Callison-Burch, 2005).

1.1.1. Thesis Statement

In this thesis, we claim that bilingually-induced paraphrases provide useful signals for computational modeling of lexical semantics. Further, these signals are complementary to information derived from monolingual distributional and pattern-based methods due to several key characteristics. First, the set of paraphrases for a polysemous word contains terms pertaining to its various senses, which enables us to use paraphrases to model word sense. Second, because the pivot method used to derive these paraphrases is rooted in phrase-based machine translation, paraphrases include both single-word terms and multi-word phrases, and thus can be used to analyze relationships between compositional phrases and their single-word paraphrases. Third, paraphrases can be extracted automatically at large scale, meaning that they have wide coverage of terms in the general domain.

In the chapters that follow, we demonstrate how information derived from bilingually-induced paraphrases can be used for three specific tasks in lexical semantics: discovering the different senses of a word, predicting the relative intensity between scalar adjectives, and generating sense-specific examples of word use. In each case, we show that the information derived from bilingually-induced paraphrase signals is complementary to monolingual signals of lexical semantics such as contextual similarity and lexico-syntactic patterns.

1.1.2. Outline of this Document

The rest of this document is organized as follows.

Chapter 2

We begin with a review of related work in the areas of paraphrasing and lexical semantics. First, we introduce the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013; Pavlick et al., 2015b), which is a resource of bilingually-induced paraphrases that is central to the rest of the thesis. Next, we review commonly exploited sources of signal for lexical semantic models, including monolingual and bilingual distributional properties of words, lexico-syntactic patterns, and sentiment. After that, we briefly review tasks in the study of lexical semantics that are related to the work presented here, including word sense induction, predicting scalar adjective intensity, and semantic relation prediction. The section concludes with a short description of two neural text representation models that are used throughout the following chapters.

Chapter 3

The terms *hot* and *dish* in our original question about Chinese food can each take on a variety of meanings, depending on the context in which they appear. For such polysemous words, the potential variation in meaning can be drastic, as in case of a homonym like *lie* with its deception and reclining senses, or the variations can be more subtle, as with the noun *dance* and its movement or social gathering senses. The different meanings of a word are reflected in the set of its paraphrases (Apidianaki et al., 2014). For example, paraphrases for the noun *coach* include *bus*, *manager*, *trainer*, *mentor*, and *carriage*, which pertain to its automobile and person senses. Applications that rely on choosing appropriate paraphrases for a given word in context, like query expansion (Maron and Kuhns, 1960) or lexical substitution (McCarthy and Navigli, 2009), must therefore incorporate a way to filter out inappropriate paraphrases for a given target word in context.

In Chapter 3, we address the task of clustering the paraphrases of a target word by the sense of the target that they convey. This task is very similar to word sense induction (WSI), which aims to discriminate the possible meanings of a target word present within a corpus

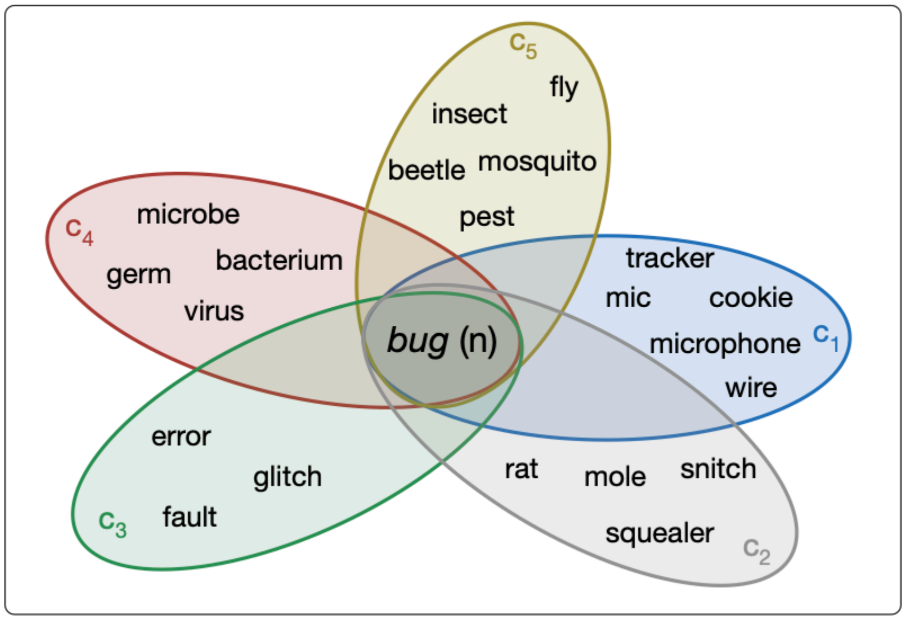


Figure 2: In Chapter 3, we cluster the paraphrases of a target word like the noun *bug* to uncover its different senses.

(Navigli, 2009). Our modeling approach in this chapter implicitly assumes that the senses of a target word can be discretely partitioned, and that these partitions can be represented by a human-generated ‘ground truth’ sense inventory (which we aim to replicate automatically). Our work aims to both validate the earlier finding of Apidianaki et al. (2014) that a target word’s paraphrases can be clustered to uncover its senses, and to examine whether signals derived from (bilingually-induced) paraphrases are as effective at discriminating word sense as signals from monolingual contextual similarity or translation overlap. In the process of generating sense clusters, these signals are used to (a) measure semantic similarity between terms being clustered, and (b) to assess cluster quality in order to choose an ‘optimal’ number of clusters or senses. Via a series of experiments, we vary the metrics used for (a) and (b), and evaluate the predicted clusters intrinsically by comparing their overlap with sets of human-generated sense clusters. The results indicate that on average, paraphrase strength out-performs the other metrics when used for measuring term similarity. However, the best clustering results are achieved by combining paraphrase strength with monolingual contextual similarity, showing that the two types of information are complementary. This work has been previously published in (Cocos and Callison-Burch, 2016).

Our clustering experiments are followed with an extrinsic evaluation and demonstration of how the resulting sense clusters can be applied to the task of lexical substitution, i.e. choosing appropriate substitutes for a word in context that retain the original meaning. Most recent lexical substitution systems ignore any explicit word sense representation when proposing substitutes, instead relying on word embedding similarity alone (Melamud et al., 2015b; Roller and Erk, 2016a; Melamud et al., 2016). In Section 3.8, we propose the method of 'sense promotion' which can be applied as a post-processing step to embedding-based (sense agnostic) systems that rank substitution candidates. Given a target word instance in context, the method simply estimates the relevance of each of the target word's paraphrase sense clusters given the context, and promotes the rank of terms belonging to the most relevant cluster. This step improves the lexical substitution performance of an existing embedding-based system by 6% using a simple baseline disambiguation method to choose the most relevant cluster, and has the potential to improve performance by up to 25% given a better performing disambiguation system. Portions of this work were reported previously in (Cocos et al., 2017).

Chapter 4

Asking for a Chinese food that is *not so hot* implies that among adjectives describing spiciness, like peppery, zesty, spicy, and fiery, there is a range of intensities. Understanding these differences is necessary to provide a good answer to the question; a Chinese dish described as *zesty* would be an appropriate answer, but one described as *like lava* would not. Chapter 4 addresses the task of predicting the relative intensity relationship between pairs of scalar adjectives that describe a shared attribute like spiciness. We propose a new paraphrase-based method to predict the relative intensity relation that holds between an adjective pair based on the idea that, for example, paraphrase pair (*really hot* $\leftrightarrow$ *fiery*)

Paraphrase pair…		…suggests that
particularly pleased	*ecstatic*	*pleased* < *ecstatic*
quite limited	*restricted*	*limited* < *restricted*
rather odd	*crazy*	*odd* < *crazy*
so silly	*dumb*	*silly* < *dumb*
completely mad	*crazy*	*mad* < *crazy*
RB JJ_1	JJ_2	$JJ_1 < JJ_2$

Figure 3: Our paraphrase-based method for predicting relative adjective intensity relies on paraphrase pairs in which an intensifying adverb (RB) and an adjective (JJ_1) are paired with a second adjective (JJ_2), indicating that the first adjective is less intense than the second.

suggests that *hot* is less intense than *fiery*. Due to the broad coverage and noise inherent in the paraphrase data, our method provides predictions for more adjective pairs at lower accuracy than methods that rely on lexico-syntactic patterns or a hand-compiled adjective intensity lexicon. We show that combining paraphrase evidence with the existing, complementary approaches improves the quality of systems for automatically ordering sets of scalar adjectives and inferring the polarity of indirect answers to yes/no questions. The content of this chapter was published in (Cocos et al., 2018b).

Chapter 5

Chapters 3 and 4 describe ways to derive signals from paraphrases that are useful for learning about aspects of computational lexical semantics, and show that these bilingually-induced signals can be combined directly with monolingual signals in a complementary way. In Chapter 5 we explore a different type of complementary relationship between paraphrases and monolingual contextual similarity. Namely, we describe a way in which paraphrases can be leveraged to automatically generate a large resource of word usages with a particular fine-grained meaning. The resulting micro-sense tagged corpus can then be used for training sense-aware models using traditional methods based on distributional properties or patterns.

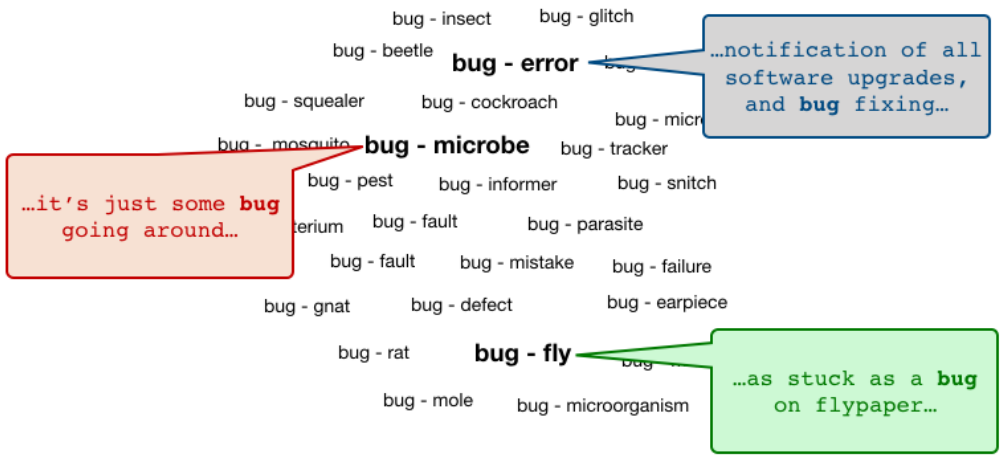


Figure 4: In Chapter 5, we apply bilingual pivoting (Bannard and Callison-Burch, 2005) to generate sentence-level contexts for paraphrases. Here we show context snippets for several different paraphrases, or fine-grained senses, of the noun *bug*.

We propose a new method for automatically enumerating example usages of a query word having a particular meaning. The method is grounded in the idea that a word's paraphrases represent its fine-grained senses, i.e. *bug* has different meanings corresponding to its paraphrases *error*, *fly*, and *microbe*. To find sentences where *bug* is used in its *error* sense, we extract sentences from bitext corpora where *bug* is aligned to a translation it shares with *error* (Figure 4).

This idea is used to automatically generate a large resource of example word usages with a particular fine-grained sense. This resource, which we call Paraphrase-Sense-Tagged Sentences (PSTS), contains up to 10k sentence-level examples for the 3 million highest-quality paraphrase pairs in PPDB. The quality of sentences in PSTS are evaluated by humans, and a re-ranking model is trained to enable selection of the highest-quality sentences for each paraphrase pair.

Chapter 6

Chapter 6 continues the work of the previous chapter by providing three examples of how PSTS can be used to train models for lexical semantic tasks where knowledge of word sense is important. We begin by using PSTS as a corpus for training fine-grained sense embeddings, where senses are instantiated by paraphrases, based on existing word representation models. The paraphrase embeddings are directly compared with their word-type embedding counterparts through an intrinsic evaluation on a battery of semantic similarity and relatedness benchmarks. The experiments show that the paraphrase embeddings trained on PSTS capture a more precise notion of semantic similarity than word-type embeddings. Next, the paraphrase embeddings are used in conjunction with sense clusters derived in Chapter 3 for word sense induction: given a target word instance, we assume that the sense clusters represent the target's available sense inventory, and map the instance back to the most appropriate sense cluster using the paraphrase embeddings. This method produces competitive results on two existing WSI datasets. Finally, we use PSTS to automatically create a large training dataset for the task of predicting hypernymy in context. To assess the quality of the training set, we fine-tune the BERT transformer encoder model (Devlin et al., 2019) for the task of contextual hypernym prediction, and evaluate the performance of this model when trained on PSTS versus an existing hand-crafted training set.

As in Chapter 3, Chapters 5-6 explore the use of paraphrases for modeling word sense. However, the approach taken is quite different and is based on a different set of underlying assumptions. In Chapter 3, we assume that the meanings of a word can be discretely partitioned and represented by means of a human-generated sense inventory. The goal of paraphrase sense clustering, then, is to automatically replicate the set of human-generated sense clusters for a target word. We then show that this model of word sense can be combined with a sense-agnostic lexical substitution model to improve performance in that task. In Chapter 5, rather than assuming that some closed set of senses exists for each word, we use paraphrases to instantiate the various possible meanings of a word. This approach is more flexible, as it is not tied to a sense inventory (although it is straightforward to map paraphrases onto a sense inventory if desired, as is done in the WSI experiment). The experiments in Chapters 3 and 6 show that both abstractions of word sense – as paraphrase clusters, versus individual paraphrases – can be useful insofar as they model variable word meaning in a way that improves performance in downstream tasks.

Chapter 7

To conclude, Chapter 7 summarizes the contributions made in this thesis, its limitations, and suggests potential areas for continued work.

CHAPTER 2 : Background and Related Work

2.1. The Paraphrase Database

The resource most central to the work in this thesis is the Paraphrase Database (PPDB)[1] (Ganitkevitch et al., 2013; Pavlick et al., 2015b), a collection containing over 220M English paraphrase pairs. Of the pairs, roughly 8M are lexical, or single-word, pairs (e.g. *marine* $\leftrightarrow$ *maritime*), 73M are phrasal, or multi-word, pairs (e.g. *marine* $\leftrightarrow$ *oceans and seas*), and 140M are pairs of syntactic patterns (e.g. *in collaboration [IN]* $\leftrightarrow$ *[IN] the cooperation of*). PPDB is distributed in a variety of sizes from S to XXXL, ranging from smallest and most precise, to largest and noisiest. Throughout this work, we use lexical and phrasal pairs from the XXL version.

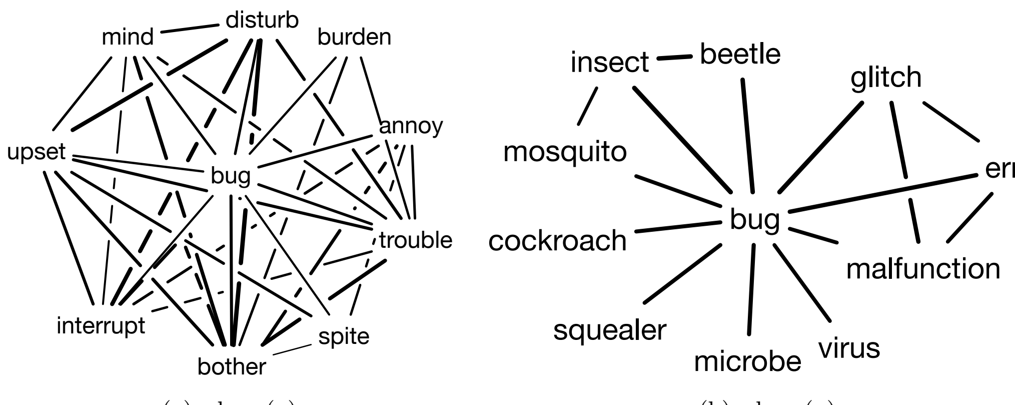


Figure 5: PPDB graphs for the verb (a) and noun (b) forms of *bug* and up to 10 of their highest-strength paraphrases, ordered by PPDBSCORE. Line width corresponds to PPDBSCORE.

PPDB contains words and phrases that are predicted to have similar meaning on the basis of their bilingual distributional similarity. Specifically, PPDB was produced automatically via the bilingual pivoting method (Bannard and Callison-Burch, 2005), which posits that if two English words e_1 and e_2 share a common foreign translation f, then this is evidence that e_1 and e_2 share similar meaning. For example, the English verb *sleep* and verb phrase

[1]`http://paraphrase.org/`

go to bed share the French translations *mettre au lit* and *bonne nuit*, indicating that they have similar semantics.

The resource was generated by applying bilingual pivoting over a corpus of more than 106M aligned English-foreign sentence pairs covering 26 pivot languages. The word alignment between parallel sentences was done automatically, which introduced noise into the pivoting process. As a result, PPDB paraphrases have varying quality. In order to rank paraphrase pairs, Pavlick et al. (2015b) introduced the PPDB 2.0 Score (hereafter PPDBSCORE), a supervised metric designed to correlate with human judgments of paraphrase quality. Since the human annotations used for training the PPDBSCORE model were based on a 1-5 Likert scale (where higher scores indicate better-quality paraphrases), the PPDBSCORE values are predicted to match this range (although a small fraction of the predictions fall slightly outside the range). For example, the paraphrase pair *marine* ↔ *maritime* has a PPDBSCORE of 3.4, while the pair *marine* ↔ *fleet* has a PPDBSCORE of 1.5.

Paraphrases in PPDB are partitioned by syntactic type following the work of Callison-Burch (2008). He showed that applying syntactic constraints during paraphrase extraction via the pivot method improves paraphrase quality. This means that a query for paraphrases of the noun *marine* will return other nouns like *crewmen* and *sea*, while a query for paraphrases of the adjective *marine* will return other adjectives like *naval, marine-based, offshore*, and others. Throughout this work, we use the term *paraphrase set* to refer to the set of PPDB paraphrases for a given query consisting of a target phrase and its part of speech (Definition 2.1.1).

Definition 2.1.1: Paraphrase Set

A *paraphrase set* (PPSET) is the unordered set of PPDB XXL paraphrases for a given query, which consists of a target phrase and its part of speech.

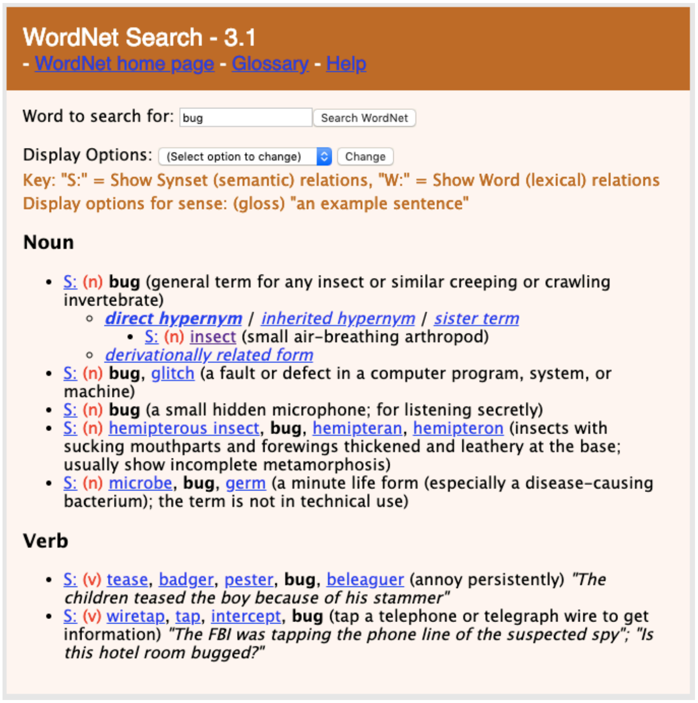
ex) $\text{PPSET}(bug, n) = \{insect, beetle, error, microbe, virus, mike, squealer, \dots\}$

While paraphrases are partitioned by part of speech, not all paraphrases for a given target word are appropriate in a given context (Apidianaki, 2016). This is for two primary reasons. First, there is no explicit sense distinction within the paraphrases of a target word; although the noun *bug* has paraphrases including *insect, glitch, microbe, virus, pest,* and *microphone*, only some of these are useful in a given context because they pertain to different meanings of *bug*. Second, the paraphrase relationship is general and under-defined, which means that some paraphrase pairs are entailing and others are not; *bug* can be replaced with its paraphrase *organism* but not necessarily with its paraphrase *mosquito* in the sentence *The scientist examined the bug under the microscope.* Pavlick et al. (2015a) took a first step toward addressing this entailment problem by locally classifying each pairwise paraphrase relation into one of six more specific entailment relations (Equivalence, Exclusion, ForwardEntailment, Independent, OtherRelated, and ReverseEntailment). But these locally-predicted entailment relations can produce logical inconsistencies when chained together; *mosquito* entails *bug* and *bug* entails *listening device*, but *mosquito* does not entail *listening device.*

2.1.1. Comparison with Other Lexical Semantic Resources

PPDB is chosen as the primary dataset for this work because its size dwarfs other paraphrase resources. However there are a number of other useful and widely-used lexical-semantic datasets, and this section gives a brief overview of several as a basis for comparison with PPDB.

One of the best known lexical semantic resources in use is the manually-compiled WordNet (Miller, 1995; Fellbaum, 1998). WordNet can be viewed as a graph, where its 117,000 nodes are "synsets" – unordered sets of synonymous lemmas – and edges represent semantic relations such as hypernymy or antonymy that exist between synsets. A given word like *bug (n)* with multiple meanings appears in one synset for each of its senses. Like PPDB, WordNet's lemmas and synsets are considered to be specific to a particular part of speech (so synsets containing the noun *bug* are distinct from synsets containing the verb *bug*). But unlike PPDB, WordNet's synset structure implicitly encodes the various possible meanings


WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: bug [Search WordNet]

Display Options: (Select option to change) [Change]
Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations
Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **bug** (general term for any insect or similar creeping or crawling invertebrate)
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) insect (small air-breathing arthropod)
  - *derivationally related form*
- S: (n) **bug**, glitch (a fault or defect in a computer program, system, or machine)
- S: (n) **bug** (a small hidden microphone; for listening secretly)
- S: (n) hemipterous insect, **bug**, hemipteran, hemipteron (insects with sucking mouthparts and forewings thickened and leathery at the base; usually show incomplete metamorphosis)
- S: (n) microbe, **bug**, germ (a minute life form (especially a disease-causing bacterium); the term is not in technical use)

Verb

- S: (v) tease, badger, pester, **bug**, beleaguer (annoy persistently) *"The children teased the boy because of his stammer"*
- S: (v) wiretap, tap, intercept, **bug** (tap a telephone or telegraph wire to get information) *"The FBI was tapping the phone line of the suspected spy"; "Is this hotel room bugged?"*

Figure 6: A screenshot of WordNet's online interface, showing the synsets for the noun and verb forms of *bug* and the hypernyms of the first synset for *bug.n* (`http://wordnetweb.princeton.edu`).

for each word type it contains. Another major difference between PPDB and WordNet is that WordNet's relation edges are specifically typed – for example, synsets containing *bug* and *flaw* are connected by a directed hypernym relation, and synsets containing *fast* and *slow* are connected by an antonym relation. As noted above, PPDB's undirected 'paraphrase' relation is overly general; each paraphrase edge in the PPDB graph could potentially be classified as a more specific type of semantic relation (Pavlick et al., 2015a). Table 1 outlines the primary differences between PPDB and WordNet, and Ganitkevitch (2018) provides a more in-depth comparison. To mitigate the issues posed by the limited size of WordNet, there is also a body of research focused on automated ways to expand its coverage (Snow et al., 2005; Yang and Callan, 2009; Navigli and Ponzetto, 2010, 2012).

While WordNet is a heavily curated and precisely defined resource, there are also other automatically-generated paraphrase resources that are closer in structure to PPDB. These include DIRT (Lin and Pantel, 2001a,b) and the Microsoft Research paraphrase tables

	PPDB XXL	WordNet	Implications
Nodes	Word types (part-of-speech tagged)	Synsets (part-of-speech tagged)	WordNet encodes polysemy; PPDB does not
Edges	Encode general 'paraphrase' relation Undirected edges	Encode specific semantic relation types (e.g. hypernymy, antonymy) Undirected for symmetric relations; directed for asymmetric relations	WordNet encodes fine-grained relations WordNet's directed relations between senses preserve *transitivity*; PPDB's undirected relations between word types do not
Coverage	255K unique word/phrase types	155K unique word/phrase types	PPDB has wider coverage of word types and general phrase relatedness
Metadata	Paraphrase probabilities; translation probabilities; PPDBSCORE; predicted entailment relations and style classification	Word frequency, synset glosses and example usage	WordNet includes curated lexicographic content, while PPDB contains mostly artefacts of the bilingual pivoting process

Table 1: Comparison between PPDB and WordNet based on structure, size, and additional information included (Metadata).

(Dolan et al., 2004; Quirk et al., 2004). Both are structured as pairs of words or phrases with similar meaning. But unlike PPDB, where paraphrases are extracted via *bilingual* pivoting, the paraphrases in DIRT and the MSR dataset are collected using *monolingual* methods. In the case of DIRT, paraphrases are extracted by finding parsed dependency paths with high distributional similarity (e.g. *"X solves Y"* $\leftrightarrow$ *"X finds a solution to Y"*). Paraphrases in the MSR dataset are extracted by finding highly similar sentences from different news stories about a particular event, and applying an automated alignment algorithm from machine translation to find meaning-equivalent phrases in the two sentences. The sizes of DIRT and the MSR datasets are both smaller than PPDB, with roughly 12M and 24M paraphrase pairs respectively.

As seen here, lexical semantic resources can be generated either automatically or manually. The choice presents a trade-off between scalability and precision or accuracy; manually-compiled resources are the most accurate (subject to the limits of human agreement) and well-defined, but are costly to update or expand to new languages and domains. Automatically-generated resources generally have wider coverage than manual resources, and can be adapted to new languages or domains with a fraction of the effort required for manual compilation. However, most automatic generation processes have some inherent noise, and thus the resulting resources may have errors or lack specificity (e.g. the relations they encode, like 'relatedness', may be under-defined). One typical paradigm is for researchers to develop automatic methods of producing lexical-semantic resources, while using existing manually-compiled resources as a basis for evaluation and tuning of the automatic generation process. Indeed that is what happens throughout this thesis. In Chapter 3, we use sense clusters constructed from WordNet synsets and as produced by crowd workers as 'ground truth' sense inventories to evaluate our automatic clustering methods. In Chapter 4, we use human-constructed adjective intensity scales to evaluate the pairwise intensity predictions produced by our model. And in Chapter 5, we rely on semantic relation datasets derived from manually-compiled resources in order to evaluate our methods for automatic relation prediction between word types.

2.2. Signals for Computational Lexical Semantics

Lexical semantics, broadly defined, concerns the meanings of and relationships between individual words. Automatic methods for learning and representing these concepts are building blocks for the long-standing goal of natural language understanding.

Computational approaches to lexical semantics focus on learning both the possible meanings of a given word (i.e. its senses), and learning to predict semantic relations that hold between a given pair of words. With respect to word senses, there is a long-running debate in the

NLP research community about how best to model a word's possible meanings. While true homonyms may have fully discrete senses (e.g. the *organization* and *weapon* senses of the noun *club*), in other cases, such as for the *computer* and *biological* senses of *virus*, the boundaries between one meaning and another are fuzzier and depend on context (Tuggy, 1993; Copestake and Briscoe, 1995; Kilgarriff, 1997; Cruse, 2000; Passonneau et al., 2010; McCarthy et al., 2016). Kilgarriff (2007) described this issue by saying, *"There are no decisive ways of identifying where one sense of a word ends and the next begins."* There have been various attempts to model word senses, from fully discrete (Miller, 1995; Navigli and Ponzetto, 2010) to fully continuous and context-dependent (Peters et al., 2018; Devlin et al., 2019). Early in the thesis, in Chapter 3, we adopt an approach to modeling word sense by clustering paraphrases which assumes a discrete underlying sense inventory; later, in Chapters 5 and 6, we take a different view of word sense that uses paraphrases to instantiate fine-grained (yet still discrete) meanings of a target word. This shift toward a finer-grained sense model reflects our view that the senses of most words are, in reality, continuous and context-dependent, although some downstream tasks such as lexical substitution and semantic similarity prediction can still benefit from discrete word sense modeling.

Relation types that are frequently studied from a computational standpoint include:

- **Hyponymy/Hypernymy**: x is a *hyponym* of y (and y is a *hypernym* of x) if x is a type of y (e.g. *mosquito* is a hyponym of *insect*).
- **Co-hyponymy**: Terms x and y are *co-hyponyms* if they share a common hypernym (e.g. *mosquito* and *ant* are co-hyponyms, with the shared hypernym *insect*).
- **Synonymy**: Synonymous terms x and y have the same meaning (e.g. *snake* and *serpent*).
- **Meronymy/Holonymy**: x is a *meronym* of y (and y is a *holonym* of x) if x is a part of y (e.g. *wing* is a meronym of *mosquito*).

- **Antonymy**: x and y are *antonyms* if they are opposite or contradictory (e.g. *hot* and *cold* are antonyms).

Computational approaches to lexical semantics aim to automatically extract some information that provides useful clues about word meaning and relationships from their observable natural environment – written text. This section gives an overview of some of the most common types of text-based signals that are used.

2.2.1. Monolingual distributional signals

The *distributional hypothesis* (Harris, 1954; Weaver, 1955; Firth, 1957; Schütze, 1992) is the foundation for much of the work in computational semantics over the past 65 years. Its premise is that words which have similar meaning tend to occur within similar contexts. Viewed another way, it suggests that we can predict how similar the meanings of two words are by comparing their contexts. This suggestion has largely borne out to be true, as evidenced by methods that vary with respect to (a) the way they define context, (b) the way they encode or represent that context, and (c) the way they measure similarity between context representations.

For measuring monolingual distributional similarity, the context of a term is usually defined as either words appearing within some pre-defined lexical neighborhood of the term (a "bag of words" model), or the term's parsed syntactic dependencies. The choice of definition is task-dependent. Dependency-based contexts lead to representations where *functionally* alike words are seen as most similar (e.g. *carpenter* and *mason*), whereas bag-of-words contexts lead to representations where words from the same *domain* are seen as most similar (e.g. *carpenter* and *wood*) (Turney, 2012; Levy and Goldberg, 2014).

Most models encode context within a vector. Where they differ is in the meaning ascribed to each of the vector's dimensions (Turney and Pantel, 2010). Some models represent a word using a sparse, high-dimensional vector space where each dimension corresponds to a specific lexical/syntactic contextual feature, or to a cluster of similar features (Brown et al., 1992). Other models reduce the dimensionality of such vectors using methods like singular value decomposition (Golub and Reinsch, 1970), principal components analysis, or latent Dirichlet allocation (Blei et al., 2003). More recently, neural "word embedding" methods have come into vogue (Bengio et al., 2003; Mikolov et al., 2013b,a; Pennington et al., 2014; Peters et al., 2018; Devlin et al., 2019). These models represent a word using a dense vector of weights from a neural model trained for some task related to language modeling, and have out-performed sparse representations on both intrinsic and extrinsic downstream tasks. In Chapter 3 we use the *skip-gram* embedding model (Mikolov et al., 2013b,a) to represent words for sense clustering, and in Chapter 5 we compare the *skip-gram* and contextualized BERT models (Devlin et al., 2019) for word representation. Both of these representation methods are described in more detail in Section 2.4.

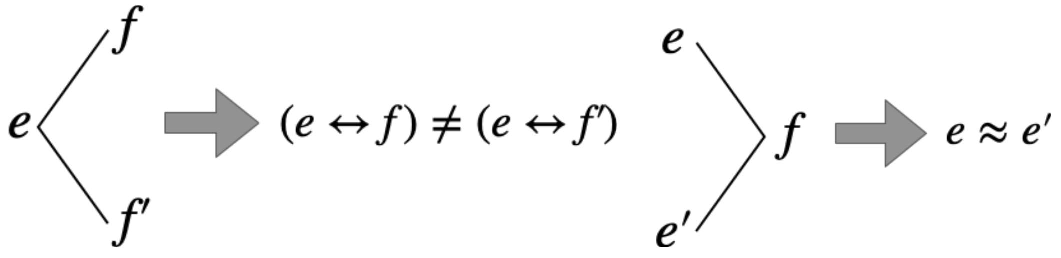
Once a term's contexts are encoded in vector form, the most common way to measure similarity (and the method that we adopt throughout this work) is via cosine similarity, which can be calculated between vectors u and v as:

$$cos(u, v) = \frac{u \cdot v}{\|u\|_2 \, \|v\|_2} \tag{2.1}$$

2.2.2. Bilingual distributional signals

If the monolingual patterns of a term provides clues about its meaning, can the same be said about its bilingual patterns? By *bilingual distributional signals*, we refer to information about words in a source language (i.e. English) that can be inferred from the translations of those words in target languages. The statistics that describe this type of information come from automatic word alignments between source and target sentences from bilingual parallel corpora.

The basic premise behind bilingual distributional signals for lexical semantics is that if two words or phrases e and e' in a source language share a foreign translation f, then one of two things can be assumed: either e and e' share meaning (the ‘synonymy’ assumption), or f is a polysemous word and the words e and e' reflect two of its senses (the ‘polysemy’ assumption). Yao et al. (2012) ran an empirical analysis of the frequency with which each assumption holds for English-Chinese and English-French parallel corpora. They found that both cases are common, with the synonymy case (when English is considered the source language) being only slightly more prevalent.



(a) Polysemy assumption (b) Synonymy Assumption

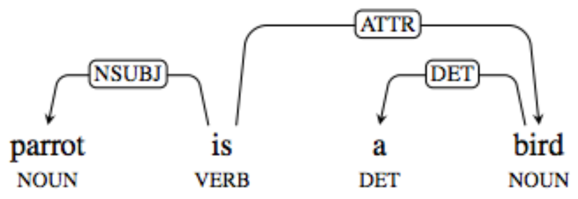
Figure 7: Viewed from an English-centric perspective, the polysemy assumption implies that if English word e aligns to different foreign words f and f', then e has two senses instantiated by its different alignments. Conversely, the synonymy assumption implies that if English words e and e' share a common foreign translation f, then e and e' have similar meaning.

Researchers have used the polysemy assumption as the basis for word sense induction and disambiguation (Brown et al., 1991; Dagan, 1991). Borrowing an example from Gale et al. (1992), if the English word *sentence* is translated to the French *peine* (judicial sentence) in one context and the French *phrase* (syntactic sentence) in another, then the two instances in English can be tagged with their appropriate senses. Most work has adopted a one-translation-per-sense modeling approach (Gale et al., 1992; Resnik and Yarowsky, 2000; Carpuat and Wu, 2007), with Carpuat and Wu (2007) going further to re-frame the task of word sense tagging as the equivalent of lexical selection in machine translation. In a related vein, Resnik and Yarowsky (2000) argued that sense inventories used for evaluation in word sense induction and disambiguation should make sense distinctions that respect translation boundaries. Apidianaki (2009a), on the other hand, argued that multiple semantically-similar translations should be clustered to represent a single sense of a target word. More generally, the polysemy assumption has been applied to automatically generating sense-tagged corpora, in order to overcome the challenges of manual sense annotation (Gale et al., 1992; Dagan and Itai, 1994; Diab and Resnik, 2002; Ng et al., 2003; Lefever et al., 2011). Recently, some work has used the polysemy assumption to generate multi-sense embeddings using cross-lingual data (Bansal et al., 2012; Guo et al., 2014; Kawakami and Dyer, 2015; Šuster et al., 2016; Upadhyay et al., 2017).

The synonymy assumption, on the other hand, has been used to find semantically related words within the same language (Dyvik, 1998; Van der Plas and Tiedemann, 2006). As we have seen, this idea can be applied to identify meaning-equivalent paraphrases using the pivot method (Bannard and Callison-Burch, 2005). Our primary dataset, PPDB, was produced this way, and this thesis aims to use paraphrases generated via bilingual pivoting as a new source of lexical semantic signal. Importantly, the pivot method can be used to identify both meaning-equivalent lexicalized words/phrases, and meaning-equivalent syntactic patterns. The latter idea has been extended to the task of sentence compression, by using bilingual pivoting to generate a synchronous tree substitution grammar, and using it to identify shorter forms of equivalent phrases (Cohn and Lapata, 2008).

2.2.3. Lexico-syntactic patterns

Pattern-based approaches identify semantic relations between pairs of terms by mining explicit patterns indicative of specific relationships from text. For example, the patterns '*X such as Y*' and '*Y, including X*' suggest the hypernym relationship *Y* IS-A *X* is likely to hold. The use of such lexico-syntactic patterns for hyponym-hypernym discovery was first suggested by Hearst (1992) and thus they are often referred to as *Hearst patterns*. Hearst patterns have also been extended to discovering PART-OF (meronymy) relations (Girju et al., 2003; Cederberg and Widdows, 2003), and relative adjective intensity (Sheinman and Tokunaga, 2009; de Melo and Bansal, 2013; Sheinman et al., 2013; Shivade et al., 2015) (e.g. the patterns "X, but not Y" and "not just X but Y" provide evidence that X is an adjective less intense than Y).



Pattern	`"X is a Y"`
Path	`X/NOUN/nsubj/<, be/VERB/ROOT/- , Y/NOUN/attr/>`

Table 2: The relation *parrot* IS-A *bird* can be represented as a Hearst pattern "*X is a Y*" or as a list of edges in the path from *parrot* to *bird* in the dependency parse.

Finding that there are cases where Hearst patterns can lead to erroneous pairs, as in the example given by Ritter et al. (2009) that breaks the pattern *X such as Y*:

> "... urban birds in cities such as pigeons ..."

subsequent work focused on improving the precision of pattern-based approaches using ranking and filtering criteria (Roark and Charniak, 1998; Cederberg and Widdows, 2003; Pantel and Ravichandran, 2004). Later work improved upon Hearst's verbatim textual pattern approach by representing patterns instead as paths from a syntactic dependency parse (Snow et al., 2005), which are less prone to errors induced by discontinuous syntactic constructions (see Table 2). In order to improve recall, these methods prioritized learned, rather than hand-crafted, syntactic dependency paths and extended the pattern mining to web scale (Paşca, 2004, 2007; Shivade et al., 2015).

Pattern-based approaches can be used either to discover semantic relations between a given set of terms, or to discover terms and relations jointly. In the latter case, pattern-based approaches can extract both entities and relations jointly via bootstrapping. Bootstrapping approaches typically take a set of hand-crafted patterns as input, and automatically discover pairs of terms matching the pattern. The discovered terms are then used to identify additional patterns indicative of the IS-A relation in an iterative manner. The pairs and patterns can be filtered at each iteration using statistical criteria to maintain quality (Riloff and Shepherd, 1997; Pantel and Pennacchiotti, 2006; Kozareva et al., 2008). Hovy et al.

(2009) report up to a seven-fold increase in the number of terms and relations discovered using bootstrapping over the number discovered using the initial hand-crafted patterns, with little drop in precision.

Patterns can also serve as features for supervised relation prediction models. In this case, a term pair (t_x, t_y) might be represented as a feature vector, where features correspond to specific patterns linking t_x and t_y in a corpus (Snow et al., 2005; de Melo and Bansal, 2013). Due to the vast lexical variability of relevant patterns, this results in a large, sparse feature space, where semantically similar patterns, such as *X be type of Y* and *X be kind of Y*, are represented independently. Nakashole et al. (2012) suggested a way of generalizing lexically similar patterns to improve recall in their PATTY system by replacing words within the pattern with part-of-speech tags or wildcards, yielding generalized patterns like *X be NOUN of Y*. But these generalized patterns can over-generalize; *X be teacher of Y* matches the generalized pattern *X be NOUN of Y*, but is not indicative of an IS-A relationship. Shwartz and Dagan (2016b) addressed this issue by embedding dependency paths using a recurrent neural network. They showed that a hypernym classification model that incorporated these embedded paths was able to learn semantically similar patterns (e.g. *X becomes Y from*) to the more generic Hearst patterns used in earlier work (e.g. *X is Y from*).

2.2.4. Sentiment

Another potential source of information about lexical semantics is the level of positive or negative *sentiment* ascribed to a span of text (Pang and Lee, 2008). For example, the statement *"I'd rather gargle battery acid than have to watch Birthday Girl again."* from Sukhdev Sandhu's review of the movie in The Daily Telegraph conveys the writer's highly negative sentiment about the movie. Conversely, a review of Yuval Harari's *Sapiens* written on Amazon by reader "Stanley," which says, *"Parts of it were downright fascinating such as 'imagination' being a keystone to human activity,"* indicates a positive view of (at least one aspect of) the book. Some sentiment is conveyed explicitly (as in the latter case) and other sentiment is implicit (as in the former).

This type of information can be particularly useful for learning about adjective polarity and intensity, based on the premise that adjectives can provide information about the sentiment of a text (Hatzivassiloglou and McKeown, 1993). When text spans are annotated either manually or automatically (e.g. via star-valued online reviews), the numeric ratings can be used as a source of information about the polarity and intensity of the adjectives contained therein (de Marneffe et al., 2010; Rill et al., 2012; Sharma et al., 2015; Ruppenhofer et al., 2014). This general idea has been used to compile *lexicons* that map adjectives to positive or negative values, such that the polarity conveys the positive or negative sentiment and the magnitude conveys the intensity. For example, a highly positive adjective like *amazing* might have a value of 5, and a slightly negative adjective like *pedestrian* might have a value of -2. In Chapter 4, we use one such lexicon, called SO-CAL (Taboada et al., 2011), as the basis for *lexicon-based* signals of adjective intensity.

2.2.5. Combining signals

Some of the lexical semantic signal types described above are complementary to one another for certain tasks. One prime example is the case of (monolingual) distributional and pattern-based methods for hypernym prediction. Because they rely on the joint appearance of two entities in text in order to make a hypernym relation prediction, pattern-based methods for entity extraction and relation prediction typically suffer from relatively low recall as compared to distributional methods (Shwartz and Dagan, 2016b). Distributional methods, on the other hand, are less precise than pattern-based methods in distinguishing hypernymy from other relation types such as equivalence, meronymy, and coordinating terms (Shwartz et al., 2017). Shwartz and Dagan (2016b) showed that combining these two types of complementary signals leads to more accurate hypernym prediction than either signal in isolation, and used this idea to produce a state-of-the-art relation prediction model called HypeNET. When trained using fully integrated path- and distributional representations of word pairs, the HypeNET model outperformed all path-based and distributional baselines by 14 F-score points.

In this thesis, we also examine ways to combine different types of lexical semantic signals for the tasks of word sense induction (Chapters 3 and 6), relative adjective intensity prediction (Chapter 4), and semantic relation prediction (Chapter 6). The following section provides a high-level overview of each of these tasks.

2.3. Lexical semantic tasks related to this work

Broadly speaking, lexical semantic tasks can be characterized as *contextual* or *non-contextual*. The distinction depends on the input to the task. In *contextual* tasks, the goal is to make a determination about the meaning of or relationships between words grounded *within a particular context*. In this case, the task input includes both word(s) about which to make a prediction, along with their surrounding context. One example of a contextual task is predicting asymmetric semantic relations in context (Shwartz and Dagan, 2016a; Vyas and Carpuat, 2017). In this setting, the system is provided with target words in two sentences, such as *The boy* ***hopped*** *toward the podium* and *The actress* ***moved*** *onstage*, and asked to determine the semantic relationship that holds between the target words (*hopped* entails *moved* in this case). In *non-contextual* tasks, systems are asked to make these predictions devoid of any context.

It is clear that the particular meaning of a word instance depends on the context in which it appears; for this reason, there is a preference to conduct lexical-semantic tasks with the benefit of added context, which provides critical information for discerning meaning. Nevertheless, there remain some cases where it may be necessary to reason about word meanings and relationships without having the benefit of context. One example is the automatic construction of taxonomies or ontologies. Another could be an information retrieval setting, where queries are frequently presented to a system devoid of context. This thesis addresses both non-contextual (Chapters 3-4) and contextual tasks (Chapter 6).

2.3.1. Word Sense Induction

Our sense clustering work in Chapter 3 is closely related to the task of word sense induction (WSI), which aims to discover all senses of a target word used within a corpus (Manandhar et al., 2010). WSI is related to, but different from, the task of word sense disambiguation (WSD), which assumes that a target word's possible senses are known a priori and aims to identify the sense used in a particular context (Navigli, 2009). This section describes four families of approaches to WSI which serve as inspiration for this work. The SEMCLUST graph clustering method, which is used as a baseline in Chapter 3, is described in more detail.

One of the most common approaches to WSI assumes that the senses of a word can be differentiated by the *monolingual* contexts in which it appears (Navigli, 2009). Namely, most instances of a particular sense of a target word will have similar neighboring words; these neighbors will be different from the neighbors of other senses of the target (e.g. the *error* sense of the target *bug* will have neighbors like *code* and *fix*, while the *organism* sense of *bug* will have neighbors like *crawl* or *winged*). Models that take this approach either frame WSI as a clustering problem, or assume a generative model. Clustering approaches aim to partition the neighbors appearing within the context of the target such that each cluster represents a distinct sense of the target word. The input to the clustering algorithm may either be a set of vector representations for each neighbor (see Figure 8), or a graph where the neighbors are nodes and edges connect similar neighbors (Schütze, 1998; Purandare and Pedersen, 2004; Bordag, 2006; Niu et al., 2007; Pedersen, 2007; Klapaftis and Manandhar, 2008). While some clustering algorithms generate a 'hard clustering' where each neighbor is partitioned into a distinct sense cluster, other approaches allow for a soft, probabilistic assignment of neighbors to clusters (Jurgens and Klapaftis, 2013), or a hierarchical clustering that reflects the categorical nature of some words' meanings (Klapaftis and Manandhar, 2008). Alternatively, the generative approach assumes that each ambiguous target word instance is drawn from a latent sense distribution, and that its neighboring context words are

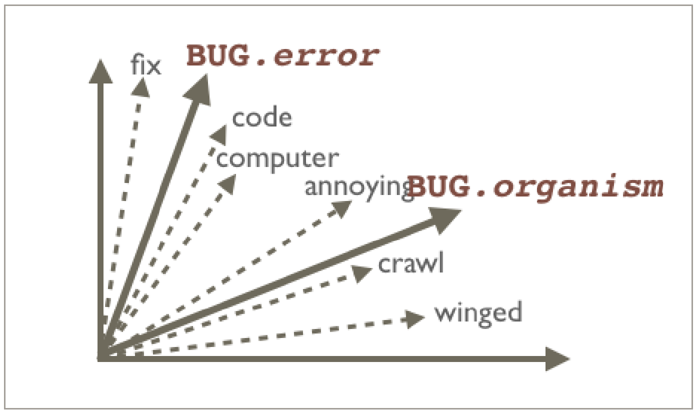


Figure 8: A toy example of context clustering for word sense induction. Contexts of the word *bug* (e.g. *fix*, *crawl*, etc) are plotted as vectors in a hypothetical two-dimensional space and partitioned into two clusters. The cluster centroids represent the *organism* and *error* senses of *bug*.

generated conditioned on the latent sense. Bayesian models are used to estimate the latent sense distribution, using the observed neighbors as evidence (Brody and Lapata, 2009; Li et al., 2010; Yao and Van Durme, 2011; Choe and Charniak, 2013).

Our proposed sense clustering method in Chapter 3 is more closely related to a second family of clustering-based WSI approaches: rather than clustering the contexts in which an ambiguous target word appears, these approaches cluster words deemed semantically similar to the target (Lin, 1998; Pantel and Lin, 2002; Dorow and Widdows, 2003; Véronis, 2004; Klapaftis and Manandhar, 2010; Hope and Keller, 2013; Pelevina et al., 2016; Panchenko et al., 2017; Ustalov et al., 2017). The intuition is that each cluster should capture a subset of the input words that pertain to a single sense of the ambiguous target. This approach has been referred to as *ego network* clustering (Pelevina et al., 2016; Panchenko et al., 2017; Ustalov et al., 2017), based on a graph encoding of the input that contains the ambiguous target word itself as the focus (ego), the set of semantically similar words to which it has some relationship (the alters), and connections between the alters (Everett and Borgatti, 2005). Our sense clustering work in Chapter 3 could be viewed as an instance of ego network clustering, where the alters consist of the target word's paraphrase set and the ego itself is removed from the graph prior to clustering.

As discussed in Section 2.2.2, bilingual distributional signals from aligned parallel corpora can provide another source of information about word sense under the polysemy assumption: if an English word like *sentence* has foreign translations *peine* (French for judicial sentence) and *phrase* (French for syntactic sentence) that are semantically different in the foreign language, this information is a clue that the English word has different meanings. This polysemy assumption has been applied to the WSI task directly by Ide et al. (2002), who clustered instances of English nouns in George Orwell's *1984* based on vectors encoding their aligned translations in six foreign language editions. They found that the groupings of word instances produced by their translation clustering method were similar to those produced by human annotators who tagged each instance with a WordNet sense. Apidianaki (2009b) take the idea of sense-tagging via foreign translations one step further; they produce clusters of the translations of English words based on their semantic similarity, such that each cluster represents a distinct sense of the English word. In Chapter 3, we take a related approach by clustering an ambiguous word's paraphrase set based on vectors of the paraphrases' translations in multiple languages.

All of the aforementioned clustering and generative approaches to WSI share the challenge that there is an unknown number of underlying senses for each ambiguous target word in a given corpus, while most clustering and Bayesian approaches require the number of clusters (k) or size of the latent space to be specified as an input parameter. Some methods proposed to circumvent this issue include adopting a non-parametric Bayesian model (Yao and Van Durme, 2011) or clustering for a range of k and choosing the optimal clustering based on some cluster quality metric (Niu et al., 2007). In Chapter 3 we take the latter approach.

The WSI work most closely related to ours is that of Apidianaki et al. (2014), who, like us, sought to determine the possible senses of a word by clustering its paraphrases. Their method (hereafter SEMCLUST) used a simple graph-based approach to cluster paraphrases on the basis of contextual similarity and shared foreign alignments. Specifically,

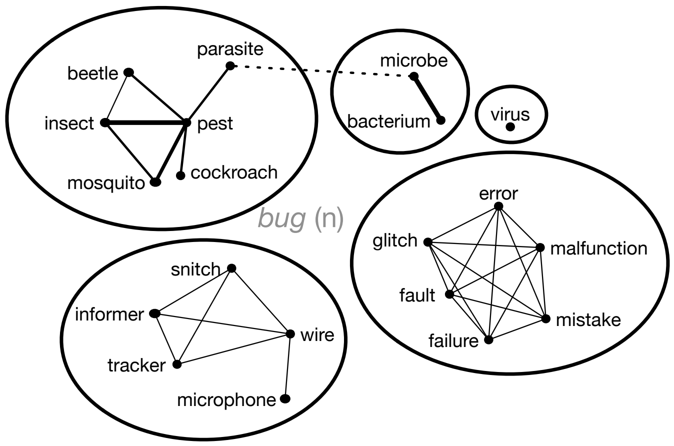


Figure 9: SEMCLUST connects all paraphrases that share foreign alignments, and cuts edges below a dynamically-tuned cutoff weight (dotted lines). The resulting connected components are its clusters.

SEMCLUST represents paraphrases as nodes in a graph and connects each pair of words sharing one or more foreign alignments with an edge weighted by contextual similarity. Concretely, for paraphrase set PPSET, it constructs a graph $G = (V, E)$ where vertices $V = \{p_i \in \text{PPSET}\}$ are words in the paraphrase set and edges connect words that share foreign word alignments in a bilingual parallel corpus. The edges of the graph are weighted based on their contextual similarity (computed over a monolingual corpus). In order to partition the graph into clusters, edges in the initial graph G with contextual similarity below a threshold T' are deleted. The connected components in the resulting graph G' are taken as the sense clusters. The threshold is dynamically tuned using an iterative procedure (Apidianaki and He, 2010).

The sense clusters induced by SEMCLUST are evaluated by comparing them to a set of reference sense clusters. These are derived from a lexical substitution dataset that groups together words which humans judge to be good substitutes for the target word in a specific context (McCarthy and Navigli, 2007). For example, a sense cluster for *figure* derived from the sentence "*The Vieth-Muller circle assumes there is angular symmetry of the corresponding points (Figure 8).*" might include the paraphrases *diagram*, *illustration*, and *picture*. Based on this evaluation, SEMCLUST outperformed simple most-frequent-sense, one-sense-per-paraphrase, and random baselines. Apidianaki et al. (2014)'s work corroborated the existence of sense distinctions in the paraphrase sets, and highlighted the need for further work to organize them by sense. In Chapter 3, we improve on their method using more advanced clustering algorithms, and by systematically exploring a wider range of similarity measures.

2.3.2. Resolving Scalar Adjective Intensity

The adjectives *warm*, *hot*, and *scalding* can all be used to describe liquid temperature, but they vary in their intensity: a coffee described as *scalding* has more extreme temperature than one described as *warm*. These types of adjectives which can be arranged along a qualitative scale are referred to as *scalar* or *gradable* adjectives. Understanding the relative intensity of adjectives that describe a common attribute has implications for sentiment analysis (Pang and Lee, 2008), question answering (de Marneffe et al., 2010), and inference (Dagan et al., 2006). Work on adjective intensity in the field of computational linguistics generally focuses on two related tasks: identifying groups of adjectives that modify a shared attribute, and ranking same-attribute adjectives by intensity. With respect to the former, common approaches involve clustering adjectives by their contexts (Hatzivassiloglou and McKeown, 1993; Shivade et al., 2015). Our work in Chapter 4 focuses on using signals from paraphrases to address the latter ranking task.

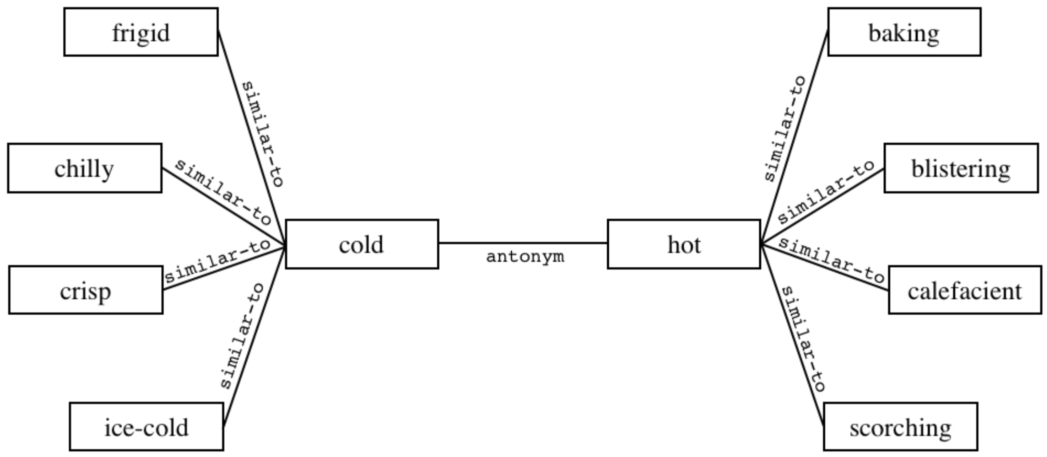


Figure 10: Example of a WordNet 'dumbbell' around the antonyms *hot* and *cold*.

Noting that adding adjective intensity relations to WordNet (Miller, 1995; Fellbaum, 1998) would be useful, Sheinman et al. (2013) propose a system for automatically extracting sets of same-attribute adjectives from WordNet 'dumbbells' – consisting of two direct antonyms at the poles and satellites of synonymous/related adjectives incident to each antonym (Figure 10) (Gross and Miller, 1990) – and ordering them by intensity. The annotations, however, are not in WordNet as of its latest version (3.1).

Existing approaches to the task of ranking scalar adjectives by their intensity mostly fall under the paradigms of *pattern-based* or *lexicon-based* approaches. *Pattern-based* approaches work by extracting lexical (Sheinman and Tokunaga, 2009; de Melo and Bansal, 2013; Sheinman et al., 2013) or syntactic (Shivade et al., 2015) patterns indicative of an intensity relationship from large corpora (see Section 2.2.3). For example, the patterns "X, but not Y" and "not just X but Y" provide evidence that X is an adjective less intense than Y.

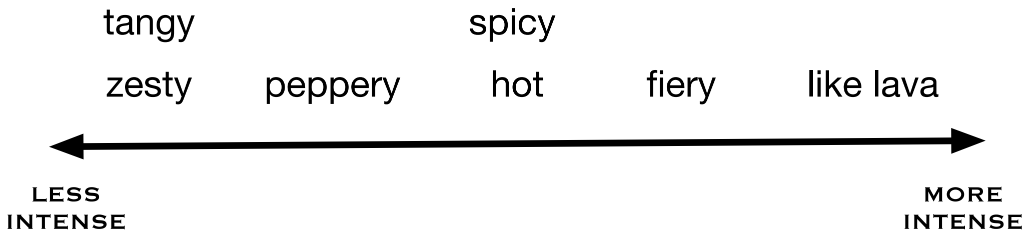


Figure 11: Scalar adjectives describing the attribute *spiciness* arranged along a hypothetical intensity scale.

Lexicon-based approaches are derived from the premise that adjectives can provide information about the sentiment of a text (Hatzivassiloglou and McKeown, 1993) (see Section 2.2.4). These methods draw upon a lexicon that maps adjectives to real-valued scores encoding both sentiment polarity and intensity. The lexicon might be compiled automatically – for example, from analyzing adjectives' appearance in star-valued product or movie reviews (de Marneffe et al., 2010; Rill et al., 2012; Sharma et al., 2015; Ruppenhofer et al., 2014) – or manually. In Chapter 4 we utilize the manually-compiled SO-CAL lexicon (Taboada et al., 2011).

Our paraphrase-based approach to inferring relative adjective intensity is based on paraphrases that combine adjectives with adverbial modifiers. A tangentially related approach is Collex (Ruppenhofer et al., 2014), which is motivated by the intuition that adjectives with extreme intensities are modified by different adverbs from adjectives with more moderate intensities: extreme adverbs like *absolutely* are more likely to modify extreme adjectives like *brilliant* than are moderate adverbs like *very.* Unlike Collex, which requires pre-determined sets of 'end-of-scale' and 'normal' adverbial modifiers, our approach learns the identity and relative importance of intensifying adverbs.

2.3.3. Semantic Relation Prediction

A major task in computational lexical semantics is determining the type of semantic relationship that holds between different words or phrases, such as *hypernymy* between *chair* and *furniture*, or *antonymy* between *hot* and *cold.* Semantic relation prediction is frequently carried out in a contextual setting as part of some larger downstream task. For example, the macro-level task of recognizing textual entailment between premise and hypothesis sentences can be decomposed into multiple (contextualized) lexical relation predictions between pairs of words or phrases aligned from the premise to the hypothesis. But because relation prediction is such a universally applicable task, it has been studied in its own right at length, and most existing benchmark datasets pose the task of predicting semantic relations between words taken out-of-context (e.g. Baroni et al. (2012); Necsulescu et al. (2015); Santus et al. (2015), and others). In Chapter 6 we work with models for both contextual (Section 6.4.4) and non-contextual (Section 6.2.3) relation prediction.

There are various types of semantic relations that are important to model as part of downstream natural language tasks. In addition to the specific named semantic relation types listed in Section 2.2 (synonymy, hypernymy, etc), fine-grained lexical entailment relations have been studied in depth (MacCartney and Manning, 2007; Pavlick et al., 2015a; Shwartz and Dagan, 2016a). A third set of commonly-studied relations concerns semantic similarity and relatedness (Rubenstein and Goodenough, 1965; Finkelstein et al., 2002; Agirre et al., 2009; Radinsky et al., 2011; Luong et al., 2013; Bruni et al., 2014; Hill et al., 2015). Predicting semantic similarity/relatedness has become increasingly common in the past few years as an intrinsic evaluation method for vector-based word representations (Baroni et al., 2014). The task setup is as follows: a system is presented with multiple pairs of words or phrases for which it must predict the degree of semantic similarity or relatedness (e.g. *snake* and *serpent* are highly similar, while *chalk* and *chalkboard* are highly related). Highly similar or related pairs get a high score, and non-similar pairs get a low score. The output of the system – typically derived from the cosine similarity of word embeddings – is compared to human judgments of semantic similarity for each pair, and systems are scored based on the correlation of their output with the human judgments.

All of the lexical semantic signals discussed in Section 2.2 can be used for relation prediction, though some signal types are better suited to particular types of relations than others (see Table 3). Here we give a brief overview of the strengths and weaknesses of each.

Relation Type	Most Predictive Signals	Least Predictive Signals
synonymy	bi-/mono-lingual distributional	patterns
antonymy	patterns	bi-/mono-lingual distributional
hypernymy	patterns, monolingual distributional	bilingual distributional

Table 3: Summary of computational signals that are best and least suited to predicting the existence of each semantic relation type versus others.

As discussed in Section 2.2.3, lexico-syntactic patterns can be used to predict relation types that are commonly expressed directly in text. These include hypernyms (pattern "*Y such as X*" suggests X is a hyponym of Y) (Hearst, 1992) and antonyms ("*either X or Y*" suggests X and Y are opposites) (Lin et al., 2003; Nguyen et al., 2017). Pattern based methods are not well suited to detecting synonyms, however, because synonyms are rarely co-located within the same short span of text.

Bilingual distributional information can also be used as a signal for relation prediction based on the synonymy assumption (Section 2.2.2). Bilingual methods are capable of finding semantically similar or related phrases, as evidenced by their use in generating PPDB (Bannard and Callison-Burch, 2005). In Pavlick et al. (2015a), the authors incorporated two types of features based on bilingual pivoting in a model for predicting fine-grained entailment relations between a pair of words or phrases. Their *paraphrase features* consisted of features distributed with PPDB, including bilingual alignment probabilities, PPDBSCORE, and lexical similarity. Their *translation features* extended inclusion measures to the overlap of foreign translations for the words in each input pair. An ablation study of their relation prediction model indicated that both the translation and paraphrase features were critical for predicting synonymy and related/unrelatedness, but were not as important for predicting hypernymy and had zero or negative impact on predicting antonymy. This finding mirrored an earlier study by Van der Plas and Tiedemann (2006), who showed that multilingual alignments from parallel corpora were more effective features for detecting synonymy than monolingual contextual features.

Monolingual distributional signals are probably the most heavily studied source of features for semantic relation prediction. It is well documented that while sharing similar monolingual contexts provides solid evidence that two terms are semantically related, it is less clear which particular semantic relationship – more specific than 'relatedness' – might hold. Word representations based on monolingual distributional signals can be tuned with specific semantic relationships like synonymy in mind (Baroni et al., 2014; Hill et al., 2015; Banjade et al., 2015). One relationship type that is notoriously difficult to distinguish using monolingual context alone is antonymy because antonymys like *fast* and *slow* tend to occur in very similar contexts (Rajana et al., 2017). For hypernymy, a thorough overview and comparison of useful distributional feature types for relation prediction is provided in Shwartz et al. (2017). They classify the types of features that can be generated from monolingual distributional signals to predict the semantic relation between a word pair as follows:

- **Similarity measures** are symmetric metrics that quantify the extent to which the monolingual contexts of each word in the pair overlap (Lin and others, 1998; Santus et al., 2016).

- **Inclusion measures** are asymmetric metrics based on the *distributional inclusion hypothesis*, which posits that for a relation t_x IS-A t_y, the contexts of the hyponym (t_x) are a subset of the contexts of the hypernym (t_y) (Weeds and Weir, 2003; Clarke, 2009; Kotlerman et al., 2010; Lenci and Benotto, 2012).

- **Informativeness measures** measure the relative *informativeness* of words in the pair, assuming that hypernyms tend to be less informative (i.e. more general) than hyponyms. The informativeness of a word can be estimated using the entropy (Santus et al., 2014) or topic coherence (Rimell, 2014) of its observed contexts.

Shwartz et al. (2017) showed that when used in an unsupervised setting, no one of these feature types was consistently better than another at hypernym prediction. Feature effectiveness varied based on the dataset, feature weighting scheme, and context type used (e.g. word-window or syntactic dependencies). In particular, certain feature types tended to be better suited for distinguishing hypernyms from other relation types, e.g. inclusion features were good at distinguishing hypernymy from meronymy, and informativeness measures reliably most distinguished hypernyms from synonyms.

Currently the best-performing distributional hypernym relation prediction models rely on fixed-dimensional, real-valued word embeddings that encode distributional semantics, such as *word2vec* (Mikolov et al., 2013b) or *Glove* (Pennington et al., 2014) (see Section 2.4). One approach is to represent an input term pair (t_x, t_y) as the concatenation (Baroni et al., 2012) or difference (Roller and Erk, 2016b; Weeds et al., 2014) of their individual word vectors, and run a supervised classification algorithm using the resulting vector as input. This approach can be viewed as discovering latent hypernymy signals encoded in the vector space. A related approach is to explicitly train word embeddings to reflect hierarchical relationships (Kruszewski et al., 2015; Vendrov et al., 2016; Li et al., 2017). Kruszewski et al. (2015) describe such a method for explicitly training word embeddings to reflect distributional inclusion in the feature space (i.e. the positive features of a hyponym are a subset of the positive features of its hypernym) and demonstrated that this out-performed the vector concatenation approach for distinguishing hypernymy from coordinate and meronym relations. A similar approach was taken by Chang et al. (2018).

One well-known property of supervised distributional hypernym prediction models is that they can succumb to *lexical memorization*, meaning that they tend to memorize prototypical hypernyms (such as *animal* and *seafood*) (Levy et al., 2015). To minimize the impact of lexical memorization, some current studies ensure that the training and test splits for hypernym prediction observe a full lexical split, meaning that no lemmas from the training set appear in the test set (Shwartz et al., 2017; Roller and Erk, 2016b).

While the previous paragraphs have addressed the use of distributional signals for predicting hypernymy out of context, distributional signals have also been extended to the task of predicting hypernymy in context. In this setting, a model is presented with two sentences c_l and c_r, each containing a target word (w_l and w_r respectively), and the task is to predict whether the target word w_l in the first sentence is a hyponym of the target w_r in the second, based on their specific meanings in the given context. Vyas and Carpuat (2017) proposed a method based on word embeddings and 'context-aware masked representations.' The general idea is to form a masked representation $w_{*,mask}$ representing each w_* within the context c_*, and then to use the masked representations $w_{l,mask}$ and $w_{r,mask}$ as input to some classification model. The masked representations $w_{*,mask}$ are formed by first generating a fixed-dimensional representation for c_*, and then taking the element-wise product ($w_* \odot c_*$) to form $w_{*,mask}$. In their paper, Vyas and Carpuat (2017) experimented with context representations created using the hidden state of a bi-directional long short-term memory recurrent neural network (Melamud et al., 2016), and via convolutional filters applied to the embeddings of the tokens in each context c_*. The performance was similar for the two methods.

2.4. Dense Word Representations

Most computational models of semantics rely on the ability to represent words or phrases in a numeric form that a computer can process. In vector space models of semantics, lexical items are represented as fixed-dimensional, real-valued numeric vectors, or *embeddings*, that encode their distribution in text. Assuming the distributional hypothesis is true, the vector for a particular term encodes its meaning such that semantically similar words have corresponding vectors that are 'close' in vector space.

While early vector space models were high-dimensional with sparse, interpretable features (corresponding to, for example, some word in a large vocabulary with which the embedded term co-occurs), most contemporary work makes use of dense, low-rank embeddings, due to their reduced size, resulting in computational efficiency and superior generalizability. There are several ways to generate dense word representations. One common approach is to apply matrix factorization methods, such as singular value decomposition (Golub and Reinsch, 1970) or principal components analysis (Pearson, 1901), to a high-dimensional matrix containing sparse word co-occurrence statistics. In the past several years, however, there has been a shift toward producing dense word representations that are an artefact of training neural models for language modeling tasks (Baroni et al., 2014). In this thesis there are two such neural representations that are used repeatedly: the *skip-gram* word embedding model (Mikolov et al., 2013b,a) and the Bidirectional Encoder Representations from Transformers (BERT) model (Devlin et al., 2019). A brief description of each follows here.

2.4.1. Skip-gram with Negative Sampling

Skip-gram word embeddings are produced as the by-product of a shallow neural model that is trained to predict the words appearing within a fixed-width context window to either side of an input target word. Stated another way, skip-gram embeddings are produced by training a model that takes a target word w_t as input, and predicts the likelihood that each other word in the model's vocabulary appears within close proximity. The training objective for a training corpus consisting of word sequence $w_1, w_2, \ldots, w_T$ is to maximize the average log likelihood of the observed neighbors of each w_t:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \tag{2.2}$$

where c is the size of the context window (an input parameter). The likelihood of seeing neighboring word w_{t+j} given target word w_t is estimated using the softmax function as:

$$p(w_{t+j}|w_t) = \frac{\exp\left({v'_{w_{t+j}}}^{\intercal} v_{w_t}\right)}{\sum_{w=1}^{|V|}\exp\left({v'_w}^{\intercal} v_{w_t}\right)} \tag{2.3}$$

Here, v_w is the input vector representation, or word embedding, for word w, and v'_w is the output vector representation, or context embedding, for word w. The quantity $|V|$ is the size of the model's vocabulary.

Training the model using Eq. 2.3 to calculate likelihood is impractical because of the cost of computing gradients for and updating all words in the vocabulary on each iteration, with typical vocabulary sizes in the range 10^5 and up. Therefore, Mikolov et al. (2013a) proposed an alternative formulation called *negative sampling* that simplifies the model while still producing high-quality word embeddings. The *skip-gram with negative sampling* (SGNS) objective uses the likelihood estimate:

$$\log p(w_{t+j}|w_t) \approx \log\sigma({v'_{w_{t+j}}}^{\intercal} v_{w_t}) + \sum_{w_k\in N}\log\sigma(-{v'_{w_k}}^{\intercal} v_{w_t}) \tag{2.4}$$

Here N is a set of words that are 'negatively sampled', i.e. randomly chosen from some underlying distribution as examples of words that do *not* appear within the context of w_t.

The size of N is an input parameter. This estimated likelihood is used instead of Eq. 2.3 in the training objective.

The SGNS model's learned parameters, therefore, are a word embedding matrix of size $|V| \times d$ containing input embeddings v_w (where d is an input parameter corresponding to the dimensionality of the resulting word embeddings), and an output projection matrix of the same size containing context embeddings. While the input embeddings are most frequently used to represent words in downstream tasks, the context embeddings can be useful as well (as in the AddCos lexical substitution model in Chapter 3 (Melamud et al., 2015b)).

One drawback of the skip-gram model is that it learns a single embedding for each word type in the vocabulary. This can be suboptimal, as many words have multiple meanings. Because skip-gram is trained using the observed contexts of each input word, the embedding for a given word tends to most closely represent its most frequent sense. As an example, consider the ten nearest skip-gram word embeddings, trained on the 4B-token Annotated Gigaword corpus, for the word *crane*:

> *cranes, backhoe, barge, forklift, ferris, winch, bulldozer, scaffold, pulley, scaffolding*

All of the nearest words based on the embedding model pertain to the construction equipment sense of *crane*, and none reflect its sense as a type of bird.

2.4.2. Bidirectional Encoder Representations from Transformers (BERT)

In order to address the issues that stem from having a single word embedding per word type, several recent papers have proposed *contextualized* word embedding models, in which the representation of a token differs depending on the context in which it appears (Peters et al., 2017, 2018; Devlin et al., 2019). One of these, called the Bidirectional Encoder Representations from Transformers (BERT) model, is featured extensively in Chapter 5.

BERT uses a multi-layer bidirectional Transformer encoder (Vaswani et al., 2017) architecture that is pre-trained on two language modeling tasks. The first, the ‘masked language model’ task, trains the model to predict masked or missing words in an input sequence. The second pre-training task is next-sentence prediction, which requires the model to determine whether the second sentence in a two-sentence sequence actually follows the first in the training corpus, or whether it has been randomly selected. The combination of the two tasks results in a model which is attuned to relationships between words within the same sentence, as well as relationships between sentences.

The BERT model takes a sequence of tokens as input, and it produces an output embedding for each token as well as an additional embedding that represents the input as a whole. The pre-trained BERT models provided by the paper's authors[2] can be fine-tuned for downstream tasks like question answering or sequence tagging. In Chapter 6 we fine-tune BERT for the task of contextualized hypernym prediction. However, it is also possible to use BERT's output token representations directly as contextualized word embeddings. We also experiment with this technique in Chapter 6.

[2]`https://github.com/google-research/bert`

CHAPTER 3 : Using Paraphrases to Induce Word Senses

3.1. Introduction

This chapter represents our first investigation into the use of paraphrases for a lexical semantic task: in this case, discovering word senses. Many natural language processing tasks rely on the ability to identify words and phrases with equivalent meaning but different wording. Automatically-generated paraphrase resources, such as PPDB, DIRT (Lin and Pantel, 2001c), and the Microsoft Research Paraphrase Phrase Tables (Dolan et al., 2004), provide instances of meaning-equivalent terms with higher coverage than manually-compiled resources like WordNet (Miller, 1995). But one drawback of these automatically generated paraphrase resources is that they group all senses of polysemous words together, and do not partition paraphrases into groups like WordNet does with its synsets. Thus a search for paraphrases of the noun *bug* would yield a single list of paraphrases that includes *insect, glitch, beetle, error, microbe, wire, cockroach, malfunction, microphone, mosquito, virus, tracker, pest, informer, snitch, parasite, bacterium, fault, mistake, failure* and many others, even though only some of these may be relevant in a particular context. The goals of this chapter are to group these paraphrases into clusters that denote the distinct senses of the target word or phrase, as shown in Figure 12, and to examine whether we can use signals from bilingually-induced paraphrases themselves to do so.

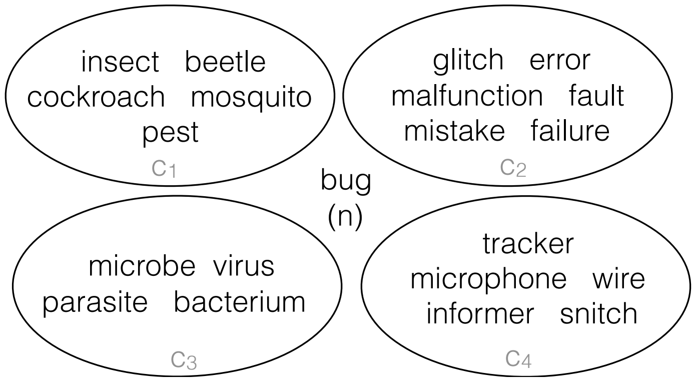


Figure 12: Our goal is to partition paraphrases of an target word like *bug* into clusters representing its distinct senses.

In the first half of this chapter, we develop a method for discriminating word sense by clustering the paraphrases from the Paraphrase Database (PPDB). Recall that PPDB contains over 100 million paraphrases generated using the bilingual pivoting method (Bannard and Callison-Burch, 2005), which posits that two English words are potential paraphrases of each other if they share one or more foreign translations. We apply two clustering algorithms, Hierarchical Graph Factorization Clustering (Yu et al., 2005; Sun and Korhonen, 2011) and Self-Tuning Spectral Clustering (Ng et al., 2001; Zelnik-Manor and Perona, 2004), and systematically explore different ways of defining the similarity matrix that they use as input. We exploit a variety of features from PPDB to cluster its paraphrases by sense, including its implicit graph structure, aligned foreign words, paraphrase scores, predicted entailment relations, and monolingual distributional similarity scores.

Our goal in the first half of this chapter is to determine which algorithm and features are the most effective for clustering paraphrases by sense. We address three research questions:

- How does paraphrase-based information compare to other similarity metrics when used for sense clustering? We systematically compare different ways of defining matrices that specify the similarity between pairs of paraphrases.

- Are better clusters produced by comparing second-order paraphrases? We use PPDB's graph structure to decide whether *mosquito* and *pest* belong to the same sense cluster by comparing lists of paraphrases for the two words.

- Can entailment relations inform sense clustering? We exploit knowledge like *beetle* is-an *insect*, and that there is no entailment between *malfunction* and *microbe*.

Our method produces sense clusters that are qualitatively and quantitatively good, and that represent a substantial improvement to the PPDB resource.

In the second part of the chapter, we demonstrate a downstream application for the induced sense clusters by using them for the task of lexical substitution. In this task, systems are presented with a target word in a sentence, and asked to propose a ranked list of appropriate substitutes that maintain the original meaning of the sentence. We show that when proposing substitutes from among the target word's PPDB paraphrases, the sense clusters can be used in conjunction with off-the-shelf embedding-based lexical substitution models to substantially improve the models' ability to propose relevant substitutes.

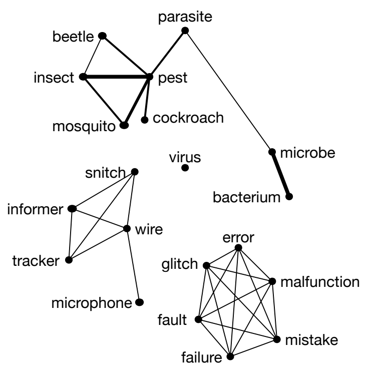
3.2. Graph Clustering Algorithms

To partition paraphrases by sense, we use two advanced graph clustering methods. Both of them allow us to experiment with a variety of similarity metrics. What follows are high-level overviews of each algorithm, succeeded by more in-depth implementation details for each method.

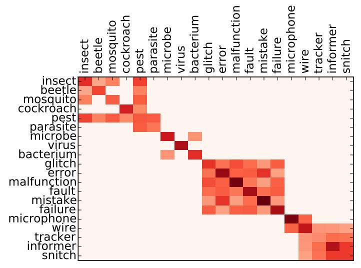
3.2.1. Hierarchical Graph Factorization Clustering

The Hierarchical Graph Factorization Clustering (HGFC) method was developed by Yu et al. (2006) to probabilistically partition data into hierarchical clusters that gradually merge fine-grained clusters into coarser ones. Sun and Korhonen (2011) applied HGFC to the task of clustering verbs into Levin (1993)-style classes. Sun and Korhonen extended the basic HGFC algorithm to automatically discover the latent tree structure in their clustering solution and incorporate prior knowledge about semantic relationships between words. They showed that HGFC far outperformed agglomerative clustering methods on their verb data set. We adopt Sun and Korhonen's implementation of HGFC for our experiments.

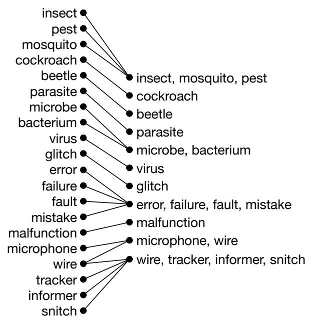
HGFC takes as input a graph $G(P, E)$, where nodes in P represent the set of paraphrases to be clustered, and undirected edges in E connect them pairwise. This graph can also be represented as a nonnegative, symmetric affinity matrix $W = \{w_{ij}\}$ where rows and columns represent paraphrases $p_i \in P$, and entries w_{ij} denote the similarity between paraphrases $sim(p_i, p_j)$. The HGFC algorithm works by iteratively factorizing W into bipartite graphs, where in the first round the nodes on the left side represent paraphrases, and nodes on the right represent senses. In subsequent rounds, nodes on the right represent coarser clustering



(a) Undirected graph for target word *bug.* Wider lines signify stronger similarity.



(b) The corresponding affinity matrix W. Darker cells signify stronger similarity.



(c) The bipartite graph induced by the first iteration of HGFC. Note *wire* is assigned to two clusters.

Figure 13: The graph, corresponding affinity matrix W, and bipartite graph created by the first iteration of HGFC for target word *bug (n)*

solutions. The output of HGFC is a set of clusterings of increasingly coarse granularity. The algorithm automatically determines the number of clusters at each level. For our task, this has the benefit that a user can choose the cluster granularity most appropriate for the downstream task (as illustrated in Figure 15). Another benefit of HGFC is that it probabilistically assigns each paraphrase to a cluster at each level of the hierarchy. If some p_i has high probability in multiple clusters, we can assign p_i to all of them (Figure 13c).

HGFC Implementation Details

This section provides further detail on the implementation of HGFC. Recall that the input to the HGFC algorithm is an affinity matrix W, where rows and columns represent paraphrases in the paraphrase set to be clustered, $p_i \in$ PPSet, and entries w_{ij} denote the similarity between paraphrases $sim(p_i, p_j)$ based on some chosen similarity measure. We achieved best results by normalizing the rows of W such that the L2 norm of each row is equal to 1.

The idea behind HGFC is that the pairwise similarity values w_{ij} can also be estimated using the construction of a bipartite graph $K(P, S)$, where one side contains paraphrase nodes p_i from P and the other consists of nodes from $S = \{s_u\}_{u=1}^{k}$ corresponding to the latent senses. Under this construction, each paraphrase in P is connected to senses in S. Specifically, the mapping from paraphrases in P to senses in S is done by the $n \times k$ affinity matrix B, where rows represent paraphrases, columns represent senses, and each matrix entry B_{iu} gives the weight between paraphrase p_i and sense s_u (Yu et al., 2005). Although paraphrase pairs are no longer directly connected in the bipartite graph, their similarity can be estimated using hops over senses $s_u \in S$:

$$w'_{ij} = \sum_{u=1}^{k} \frac{b_{iu} b_{ju}}{\lambda_u} = \left(B\Lambda^{-1}B^T\right)_{ij} \tag{3.1}$$

Here, $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_k)$ and λ_u denotes the degree of each sense vertex s_u ($\lambda_u = \sum_{i=1}^{n} b_{iu}$). If the sum of each paraphrase's row in B is 1, then intuitively b_{iu} corresponds to the likelihood that paraphrase p_i belongs to sense s_u. HGFC uses these likelihoods to produce a soft clustering from the paraphrases in P to the senses in S (Zhou et al., 2004).

HGFC uncovers B and Λ by decoupling them with $H = B\Lambda^{-1}$ and minimizing distance function $\ell(W, H\Lambda H^T)$, which gives the difference between the actual similarities in W and the estimated similarities in $H\Lambda H^T$.

Using the divergence distance $\ell(X, Y) = \sum_{ij}(x_{ij} log \frac{x_{ij}}{y_{ij}} - x_{ij} + y_{ij})$, Yu et al. (2006) showed that the following update equations are non-increasing:

$$\tilde{h}_{iu} \propto h_{iu} \sum_j \frac{w_{ij}}{(H\Lambda H^T)_{ij}} \lambda_u h_{ju}; \text{ normalize s.t.} \sum_i \tilde{h}_{iu} = 1 \tag{3.2}$$

$$\tilde{\lambda}_u \propto \lambda_u \sum_{ij} \frac{w_{ij}}{(H\Lambda H^T)_{ij}} h_{iu} h_{ju}; \text{ normalize s.t.} \sum_u \tilde{\lambda}_u = \sum_{ij} w_{ij}. \tag{3.3}$$

Finally, having minimized $\ell(W, H\Lambda H^T)$, we can calculate the new affinity matrix $\tilde{W}$ that gives affinities between senses:

$$\tilde{w}_{uv} = \sum_{i=1}^{n} \frac{b_{iu} b_{iv}}{d_i} = (B^T D^{-1} B)_{uv} \tag{3.4}$$

where $D = \text{diag}(d_1, \ldots, d_n)$ and $d_i = \sum_{u=1}^{k} b_{iu}$.

HGFC works iteratively to create clusters of increasingly coarse granularity. In each round l, the previous round's graph $\tilde{W}_{l-1}$ of size $m_{l-1} \times m_{l-1}$ is clustered into m_1 senses using equations 3.2 to 3.4. At each level l, the cluster assignment probabilities for the original $p_i \in P$ can be recovered from B_l as follows:

$$prob(s_u^{(l)}|p_i) = (D_1^{-1} B_1 D_2^{-1} B_2 D_3^{-1} B_3 \ldots D_l^{-1} B_l)_{iu} \tag{3.5}$$

We let the algorithm automatically discover the clustering tree structure by setting m_l equal to the number of non-empty clusters from round $l - 1$ minus one.

Algorithm 1 HGFC Algorithm (Yu et al. 2006)

```
Require: Paraphrase set PPSet of size $n$, affinity matrix $W$ of size $n \times n$
 1: $W_0 \leftarrow$normalize$(W)$
 2: Build the graph $G_0$ from $W_0$, and $m_0 \leftarrow n$
 3: $l \leftarrow 1$
 4: Initialize cluster count $c \leftarrow n$
 5: while $c > 1$ do
 6:     $m_l \leftarrow clustercount - 1$
 7:     Factorize $G_{l-1}$ to obtain bipartite graph $K_l$ with the affinity matrix $B_l$ of size $m_{l-1} \times$
    $m_l$ (eq. 2, 3)
 8:     Build graph $G_l$ with affinity matrix $\tilde{W}_l = B_l^T D_l^{-1} B_l$, where $D_l$'s diagonal entries are
    obtained by summation over $B_l$'s columns (eq. 4)
 9:     Compute the cluster assignment probabilities $T_l = D_1^{-1} B_1 D_2^{-1} B_2 \ldots D_l^{-1} B_l$ (eq. 5)
10:     Set $c$ equal to the number of non-empty clusters in $T$ minus one.
```

Running the HGFC algorithm returns a set of clusterings of increasingly coarse granularity. For each cluster assignment probability matrix T_l we can recover the soft clustering assignment for each input paraphrase p_i using a threshold parameter τ. We simply take the assignment for each p_i to be the set of senses with probability less than τ away from the maximum probability for that p_i, i.e. $\{s_u | abs(T_{iu}^{(l)} - max_v T_{iv}^{(l)}) \leq \tau\}$

3.2.2. Spectral Clustering

The second clustering algorithm experimented with is Self-Tuning Spectral Clustering (Zelnik-Manor and Perona, 2004). Like HGFC, spectral clustering takes an affinity matrix W as input, but the similarities end there. Whereas HGFC produces a hierarchical clustering, spectral clustering produces a flat clustering with k clusters, with k specified at runtime. The Zelnik-Manor and Perona (2004) self-tuning method is based on Ng et al. (2001)'s spectral clustering algorithm, which computes a normalized Laplacian matrix L from the input W, and executes K-means on the largest k eigenvectors of L.

Spectral Clustering Implementation Details

The algorithm is 'self-tuning' in that it enables clustering of data that is distributed according to different scales. For each data point p_i (i.e. each row in W) input to the algorithm, it constructs a local scaling parameter σ_i:

$$\sigma_i = sim(p_i, p_K) \tag{3.6}$$

where p_K is the K^{th} nearest neighbor of point p_i. Like Zelnik and Perona, we use $K = 7$ in our experiments.

Using local σ_i, we can then calculate an updated affinity matrix $\hat{A}$ based on similarities given in the input W as follows:

$$\hat{A}_{ij} = \begin{cases} \frac{w_{ij}}{\sigma_i \sigma_j} & i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

The complete algorithm we use for spectral clustering is described in Algorithm 2.

Algorithm 2 Spectral Clustering Algorithm (Ng et al. 2001, Zelnik-Manor and Perona 2004)

```
Require: Paraphrase set PPSet of size $n$, affinity matrix $W$ of size $n \times n$, number of clusters $k$
1: Compute the local scale $\sigma_i$ for each paraphrase $p_i \in$ PPSet using Eq. 3.6
2: Form the locally scalled affinity matrix $\hat{A}$, where $\hat{A}_{ij}$ is defined according to Eq. 3.7
3: Define $D$ to be a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} \hat{A}_{ij}$ and construct the normalized affinity matrix $L = D^{-1/2}\hat{A}D^{-1/2}$.
4: Find $x_1, \ldots, x_k$, the $k$ largest eigenvectors of $L$, and form the matrix $X = [x_1, \ldots, x_k] \in \mathbb{R}^{n \times k}$.
5: Re-normalize the rows of $X$ to have unit length yielding $Y \in \mathbb{R}^{n \times K}$.
6: Treat each row of $Y$ as a point in $\mathbb{R}^k$ and cluster via k-means.
7: Assign the original point $p_i$ to cluster $c$ if and only if the corresponding row $i$ of the matrix $Y$ was assigned to cluster $c$.
```

3.3. Similarity Measures

Each of our clustering algorithms take as input an affinity matrix W where the entries w_{ij} correspond to some measure of similarity between words i and j. For the 20 paraphrases in Figure 12, W is a 20x20 matrix that specifies the similarity of every pair of paraphrases like *microbe* and *bacterium* or *microbe* and *malfunction.* We systematically investigated four types of similarity scores to populate W.

3.3.1. Paraphrase Scores

Bannard and Callison-Burch (2005) defined a *paraphrase probability* in order to quantify the goodness of a pair of paraphrases, based on the underlying translation probabilities used by the bilingual pivoting method. Recall from Section 2.1 that more recently, (Pavlick et al., 2015b) used supervised logistic regression to combine a variety of scores so that they align with human judgements of paraphrase quality. PPDB 2.0 provides this nonnegative, real-valued PPDBSCORE for each pair of words in the database, although the scores are not necessarily symmetric (i.e. PPDBSCORE(i, j) may not equal PPDBSCORE(j, i)). It can be used directly as a similarity measure:

$$w_{ij} = \begin{cases} \max(\textsc{ppdbscore}(i,j), \textsc{ppdbscore}(j,i)) & (i,j) \in \text{PPDB} \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

In our dataset, values for PPDBSCORE range from 1.3 to 5.6. PPDB 2.0 does not provide a score for a word with itself, so we set PPDBSCORE(i,i) to be the maximum PPDBSCORE(i,j) such that i and j have the same stem. The PPDBSCORE for word pairs that are not linked in PPDB defaults to 0.

3.3.2. Second-Order Paraphrase Scores

A more recent family of approaches to WSI represents a word as a feature vector of its substitutable words, i.e. paraphrases (Yatbaz et al., 2012; Baskaya et al., 2013; Melamud et al., 2015a).

Work by Baskaya et al. (2013) and Melamud et al. (2015a) showed that comparing words on the basis of their *shared* paraphrases is effective for WSI. We define two novel similarity metrics that calculate the similarity of words i and j by comparing their second-order paraphrases. Instead of comparing *microbe* and *bacterium* directly with their PPDB 2.0 score, we look up all of the paraphrases of *microbe* and all of the paraphrases of *bacterium*, and compare those two lists.

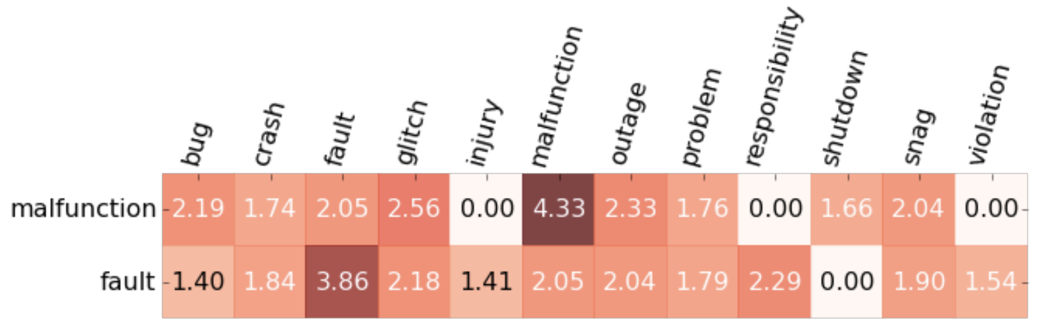


Figure 14: Comparing second-order paraphrases for *malfunction* and *fault* based on *word-paraphrase* vectors. The value of vector element v_{ij} is PPDBSCORE(i,j).

Specifically, we form notional *word-paraphrase* feature vectors v_i^p and v_j^p where the features correspond to words with which each is connected in PPDB, and the value of the k^{th} element of v_i^p equals PPDBSCORE(i, k). We can then calculate the cosine similarity or Jensen-Shannon divergence between vectors:

$$sim_{PPDB.cos}(i, j) = cos(v_i^p, v_j^p) \tag{3.9}$$

$$sim_{PPDB.js}(i, j) = 1 - JS(v_i^p, v_j^p) \tag{3.10}$$

where $JS(v_i^p, v_j^p)$ is calculated assuming that the paraphrase probability distribution for word i is given by its L1-normalized *word-paraphrase* vector v_i^p. Concretely, the Jensen-Shannon divergence is given by:

$$JS(v_i^p, v_j^p) = \frac{1}{2}KL(v_i^p \parallel M) + \frac{1}{2}KL(v_j^p \parallel M) \tag{3.11}$$

where KL is Kullback-Liebler divergence and $M = \frac{1}{2}(v_i^p + v_j^p)$.

3.3.3. Similarity of Foreign Word Alignments

Like earlier methods that use multilingual word alignments from parallel corpora to approximate the semantic similarity of English words or word instances (Dyvik, 1998; Ide et al., 2002; Van der Plas and Tiedemann, 2006), we implement a third similarity metric that estimates word similarity based on foreign alignments.

PPDB is derived from bilingual corpora. We recover the aligned foreign words and their associated translation probabilities that underlie each PPDB entry. For each English word in our dataset, we get each foreign word that it aligns to in the Spanish and Chinese bilingual parallel corpora used by Ganitkevitch and Callison-Burch (2014). We use this to define a foreign word alignment similarity metric, $sim_{TRANS}(i, j)$ for two English paraphrases i and

j. This is calculated as the cosine similarity of the *word-alignment* vectors v_i^a and v_j^a where each feature in v^a is a foreign word to which i or j aligns, and the value of entry v_{if}^a is the translation probability $p(f|i)$.

$$sim_{TRANS}(i,j) = cos(v_i^a, v_j^a) \tag{3.12}$$

In our work we use Spanish and Chinese foreign translations and probabilities drawn from the corpora used to generate the Multilingual PPDB (Ganitkevitch and Callison-Burch, 2014).

3.3.4. Monolingual Distributional Similarity

Lastly, we populate the affinity with a distributional similarity measure based on WORD2VEC (Mikolov et al., 2013b). Each paraphrase i in our data set is represented as a 300-dimensional WORD2VEC embedding v_i^w trained on part of the Google News dataset.[1] Phrasal paraphrases that did not have an entry in the WORD2VEC dataset are represented as the mean of their individual word vectors, and we use a rule-based method to map British to Americanized spellings where necessary. We use the cosine similarity between WORD2VEC embeddings as our measure of distributional similarity.

$$sim_{DISTRIB}(i,j) = cos(v_i^w, v_j^w) \tag{3.13}$$

3.4. Determining the Number of Senses

The optimal number of clusters for a set of paraphrases will vary depending on how many senses there ought to be for an target word like *bug*. It is generally recognized that optimal sense granularity depends on the application (Kilgarriff, 1997; Palmer et al., 2007; Ide and Wilks, 2007). WordNet has notoriously fine-grained senses, whereas most word sense disam-

[1] `https://code.google.com/p/word2vec/`

biguation systems achieve better performance when using coarse-grained sense inventories (Navigli, 2009). Depending on the task, the sense clustering for target word *coach* in Figure 15b with $k = 5$ clusters may be preferable to the alternative with $k = 3$ clusters. An ideal algorithm for our task would enable clustering at varying levels of granularity to support different downstream NLP applications.

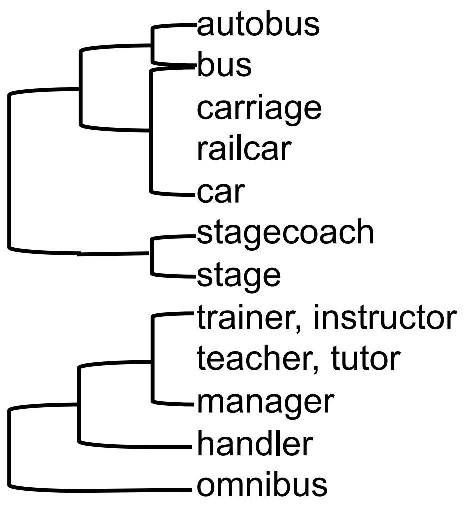
Both of our clustering algorithms can produce sense clusters at varying granularities. To determine the optimal number of clusters for a given input and clustering algorithm, we use the mean Silhouette Coefficient (Rousseeuw, 1987) to measure the 'quality' of various clustering solutions at different levels of granularity. The Silhouette Coefficient balances optimal inter-cluster tightness and intra-cluster distance, and is calculated for each paraphrase p_i as

$$s(p_i) = \frac{b(p_i) - a(p_i)}{max\{a(p_i), b(p_i)\}} \tag{3.14}$$

where $a(p_i)$ is p_i's average intra-cluster distance (average distance from p_i to each other p_j in the same cluster), and $b(p_i)$ is p_i's lowest average inter-cluster distance (distance from p_i to the nearest external cluster centroid). The Silhouette Coefficient calculation takes as input a matrix of pairwise distances, so we simply use $1 - W$ where the affinity matrix W is calculated using one of the similarity methods we previously defined.

For each clustering algorithm, we choose as the 'solution' the clustering which produces the highest mean Silhouette Coefficient. For HGFC this requires calculating the mean Silhouette Coefficient at each level of the resulting tree structure and choosing the level that maximizes the score. For spectral clustering, where the number of clusters must be specified prior to execution, we cluster each paraphrase set for a range of cluster numbers $k \in [2, min(20, n]$, where n is the number of paraphrases, and choose the optimal solution based on mean Silhouette Coefficient.[2]

[2]For spectral clustering there has been significant study into methods for automatically determining the optimal number of clusters, including analysis of eigenvalues of the graph Laplacian, and finding the rotation

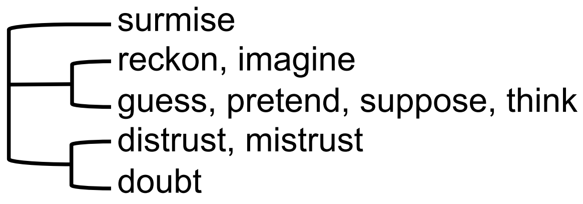


(a) HGFC clustering result for *coach* (n)

k=5
c_1: trainer, tutor, instructor, teacher
c_2: stagecoach, stage
c_3: omnibus, bus, autobus
c_4: car, carriage, railcar
c_5: manager, handler

k=3
c_1: trainer, tutor, instructor, teacher, manager, handler
c_2: stagecoach, stage
c_3: omnibus, bus, autobus, car, carriage, railcar

(b) Spectral clustering result for *coach* (n)



(c) HGFC clustering results for *suspect* (v)

k=3
c_1: reckon, pretend, think, imagine
c_2: guess, suppose, surmise
c_3: distrust, doubt, mistrust

k=5
c_1: reckon, think
c_2: pretend, imagine
c_3: guess, doubt
c_4: suppose, surmise
c_5: distrust, mistrust

(d) Spectral clustering results for *suspect* (v)

Figure 15: HGFC and Spectral Clustering results for *coach (n)* and *suspect (v)*.

3.5. Incorporating Entailment Relations

Pavlick et al. (2015a) added a set of automatically predicted semantic entailment relations for each entry in PPDB 2.0. The entailment types that they include are *Equivalent, Forward Entailment, Reverse Entailment, Exclusive*, and *Independent.* While a negative entailment relationship (*Exclusive* or *Independent*) does not preclude words from belonging to the same sense of some target word, a positive entailment relationship (*Equivalent, Forward/Reverse Entailment*) does give a strong indication that the words belong to the same sense.

of the Laplacian that brings it closest to block-diagonal (Zelnik-Manor and Perona, 2004). We experimented with these and other cluster analysis methods such as the Dunn Index (Dunn, 1973) in our work, but found that using the simple Silhouette Coefficient produced clusterings that were competitive with the more intensive methods, in far less time.

We seek a straightforward way to determine whether entailment relations provide information that is useful to the final clustering algorithm. Both of our algorithms take an affinity matrix W as input, so we add entailment information by simply multiplying each pairwise entry by its entailment probability. Specifically, we set

$$w_{ij} = \begin{cases} (1 - p_{ind}(i,j))sim(i,j) & (i,j) \in \text{PPDB} \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

where $p_{ind}(i,j)$ gives the PPDB 2.0 probability that there is an $Independent$ entailment relationship between words i and j. Intuitively, this should increase the similarity of words that are very likely to be entailing like *fault* and *failure*, and decrease the similarity of non-entailing words like *cockroach* and *microphone*.

3.6. Experimental Setup

We follow the experimental setup of Apidianaki et al. (2014). We focus our evaluation on a set of target words drawn from the LexSub test data (McCarthy and Navigli, 2007), plus 16 additional handpicked polysemous words.

3.6.1. Gold Standard Clusters

One challenge in creating our clustering methodology is that there is no reliable PPDB-sized standard against which to assess our results. WordNet synsets provide a well-vetted basis for comparison, but only allow us to evaluate our method on the 38% of our PPDB dataset that overlaps it. We therefore evaluate performance on two test sets. Examples of clusters from each dataset are given in Appendix A.3.

WordNet+ Our first test set is designed to assess how well our solution clusters align with WordNet synsets. We chose 185 polysemous words from the SEMEVAL 2007 dataset and an additional 16 handpicked polysemous words. For each we formed a paraphrase set that was the intersection of their PPDB 2.0 XXXL paraphrases with their WordNet synsets, and their immediate hyponyms and hypernyms. Each reference cluster consisted of a WordNet synset, plus the hypernyms and hyponyms of words in that synset. On average there are 7.2 reference clusters per paraphrase set.

CrowdClusters Because the coverage of WordNet is small compared to PPDB, and because WordNet synsets are very fine-grained, we wanted to create a dataset that would test the performance of our clustering algorithm against large, noisy paraphrase sets and coarse clusters. For this purpose we randomly selected 80 target words from the SEMEVAL 2007 dataset and created paraphrase sets from their unfiltered PPDB2.0 XXL entries. We then iteratively organized each paraphrase set into reference senses with the help of crowd workers on Amazon Mechanical Turk. On average there are 4.0 reference clusters per paraphrase set. A full description of our method is included in Appendix A.2.

3.6.2. Evaluation Metrics

We evaluate our method using two standard metrics: the paired F-Score and V-Measure (see Appendix A.1). Both were used in the 2010 SemEval Word Sense Induction Task (Manandhar et al., 2010) and by Apidianaki et al. (2014). We give our results in terms of weighted average performance on these metrics, where the score for each individual paraphrase set is weighted by the number of reference clusters for that target word.

3.6.3. Baselines

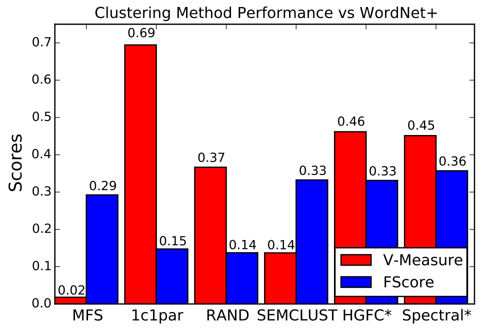
We evaluate the performance of HGFC on each dataset against the following baselines:

Most Frequent Sense (MFS) assigns all paraphrases $p_i \in P$ to a single cluster. By definition, the completeness of the MFS clustering is 1.

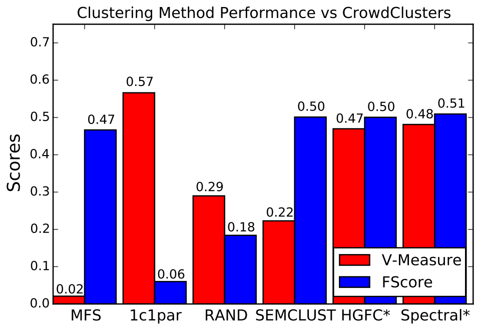
One Cluster per Paraphrase (1c1par) assigns each paraphrase $p_i \in P$ to its own cluster. By definition, the homogeneity of 1c1par clustering is 1.

Random (RAND) For each query term's paraphrase set, we generate five random clusterings of $k = 5$ clusters. We then take F-Score and V-Measure as the average of each metric calculated over the five random clusterings.

SEMCLUST We implement the SEMCLUST algorithm (Apidianaki et al., 2014) (Section 2.3.1) as a state-of-the-art baseline. Since PPDB contains only pairs of words that share a foreign word alignment, in our implementation we connect paraphrase words with an edge if the pair appears in PPDB. We adopt the WORD2VEC distributional similarity score $sim_{DISTRIB}$ for our edge weights.



(a) Clustering method performance against WordNet+



(b) Clustering method performance against CrowdClusters

Figure 16: Hierarchical Graph Factorization Clustering and Spectral Clustering both significantly outperform all baselines except 1C1PAR V-Measure.

3.7. Experimental Results

Figure 16 shows the performance of the two advanced clustering algorithms against the baselines. Our best configurations[3] for HGFC and Spectral outperformed all baselines except 1C1PAR V-Measure, which is biased toward solutions with many small clusters (Manandhar et al., 2010), and performed only marginally better than SEMCLUST in terms of F-Score alone. The dominance of 1C1PAR V-Measure is greater for the WordNet+ dataset which has smaller reference clusters than CrowdClusters. Qualitatively, we find that methods

[3]Our top-scoring Spectral method, Spectral*, uses entailments, PPDBSCORE similarities, and $sim_{DISTRIB}$ to choose k. Our best HGFC method, HGFC*, uses entailments, $sim_{DISTRIB}$ similarities, and PPDBSCORE to choose k.

Method	F-Score	V-Measure	Avg # Clusters
PPDBSCORE	0.410	0.437	6.0
$sim_{DISTRIB}$	0.376	0.440	5.7
$sim_{PPDB.cos}$	0.389	0.428	7.20
$sim_{PPDB.JS}$	0.385	0.425	7.1
sim_{TRANS}	0.358	0.375	6.2
SEMCLUST	0.417	0.180	2.3
Reference	1.0	1.0	5.6

Table 4: Average performance and number of clusters produced by our different similarity methods.

that strike a balance between high F-Score and high V-Measure tend to produce the 'best' clusters by human judgement. If we consider the average of F-Score and V-Measure as a comprehensive performance measure, our methods outperform all baselines.

On our dataset, the state-of-the-art SEMCLUST baseline tended to lump many senses of the target word together, and produced scores lower than in the original work. We attribute this to the fact that the original work extracted paraphrases from Europarl, which is much smaller than PPDB, and thus created affinity matrices W which were sparser than those produced by our method. Directly applied, SEMCLUST works well on small data sets, but does not scale well to the larger, noisier PPDB data. More advanced graph-based clustering methods produce better sense clusters for PPDB.

The first question we sought to address with this work was which similarity metric is the best for sense clustering. Table 4 reports the average F-Score and V-Measure across 40 test configurations for each similarity calculation method. On average across test sets and clustering algorithms, the paraphrase similarity score (PPDBSCORE) performs better than monolingual distributional similarity ($sim_{DISTRIB}$) in terms of F-Score, but the results are reversed for V-Measure. This is also shown in the best HGFC and Spectral configurations, where the two similarity scores are swapped between them.

Next, we investigated whether comparing second-order paraphrases would produce better clusters than simply using PPDBSCORE directly. Table 4 also compares the two methods that we had for computing the similarity of second order paraphrases – cosine similarity ($sim_{PPDB.cos}$) and Jensen-Shannon divergence ($sim_{PPDB.JS}$). On average across test sets and clustering algorithms, using the direct paraphrase score gives stronger V-Measure and F-score than the second-order methods. It also produces coarser clusters than the second-order PPDB similarity methods.

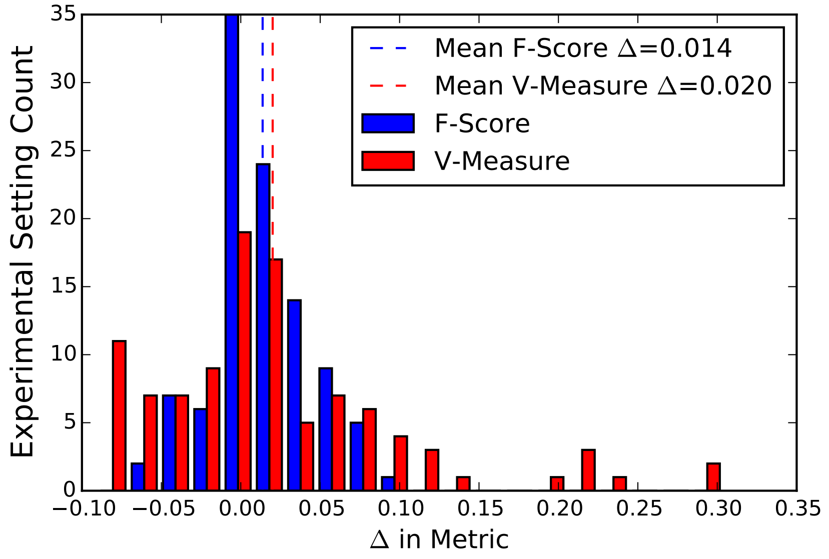


Figure 17: Histogram of metric change by adding entailment information across all experiments.

Finally, we investigated whether incorporating automatically predicted entailment relations would improve cluster quality, and we found that it did. All other things being equal, adding entailment information increases F-Score by .014 and V-Measure by .020 on average (Figure 17). Adding entailment information had the greatest improvement to HGFC methods with $sim_{DISTRIB}$ similarities, where it improved F-Score by an average of .03 and V-Measure by an average of .05.

3.8. Using sense clusters to improve lexical substitution

Having demonstrated how to organize paraphrase pairs into clusters denoting senses, we next show how the resulting clusters can be used as part of the downstream task of lexical substitution – i.e. choosing appropriate substitutes for a target word in context. Similarly to Apidianaki (2016), we focus on the specific goal of selecting appropriate substitutes from the set of PPDB paraphrases for the target word instance.

The lexical substitution task (McCarthy and Navigli, 2007) (hereafter 'lexsub') requires systems to predict substitutes for target word instances that preserve their meaning in context. Systems are evaluated on the extent to which the substitute ranking produced by the model agrees with a substitute ranking produced by humans. State-of-the-art lexsub models (Melamud et al., 2015b; Roller and Erk, 2016a; Melamud et al., 2016) typically use word and context embeddings to propose substitutes that are similar to the target word and a good fit for the context; they do not explicitly consider word sense when proposing substitutes.

Here we propose *sense promotion* with paraphrase sense clusters as a post-processing step that improves the lexsub results of embedding-based models. Given a substitute ranking produced by any lexsub model, sense promotion boosts the rank of words belonging to the sense cluster that is most applicable to the target context, while maintaining their relative order. In this section, sense promotion is demonstrated with PPDB sense clusters produced using the spectral clustering method described in Section 3.2.2. We show via an oracle experiment that this technique has the potential to improve the top performing embedding-based lexsub models' precision-at-5 by up to 48%, and that using a simple word sense disambiguation method to choose the most appropriate cluster we can realize gains of up to 19% over the embedding-based models' original rankings.

3.8.1. Sense Promotion for Lexical Substitution

The lexsub task presents systems with a sentence having one token designated as the *target word*. Systems are required to propose ranked substitutes for the target word in context that preserve the original meaning of the sentence. For example, given the sentence *What a funny joke!*, a good system might propose the substitute ranking *humorous, comical, silly,* *entertaining*. Lexical substitution systems are evaluated on how well their proposed ranking overlaps with a set of human-proposed substitutes. In an 'all-words' ranking setting, where the model's proposed substitutes come from an open vocabulary, the usual evaluation metric is precision-at-x (we use $x \in \{1, 3, 5\}$), that is, the proportion of the model's top-x ranked substitutes that appear in the (unordered) set of human-annotated substitutes.

Current embedding-based lexsub systems predict substitutes that are (a) similar to the target word, and (b) a good fit within the target context, on the basis of their word embedding similarity. These models typically utilize word embeddings trained to encode distributional similarity, such as *word2vec* (Mikolov et al., 2013b) or Glove (Pennington et al., 2014); they ignore explicit sense representation.

We propose *sense promotion* as a method for improving the rankings of embedding-based lexsub systems. The general idea behind sense promotion is to start with a set of ranked substitutes generated by an embedding-based model. Then, using the paraphrase sense clusters generated in this chapter as a sense inventory, we determine which cluster represents the most likely sense of the word in context using a simple word sense disambiguation method described in Section 3.8.2. Finally we elevate the relative rank of substitutes that belong to the chosen cluster to the top of the rankings. See Table 5 as an example.

In practice, sense promotion is implemented as follows. For each sentence, the set of paraphrase sense clusters for the target word is assumed as that word's sense inventory. Given a set of ranked substitutes for the target word in context as output by some embedding-based lexsub model, we simply add a large number (10,000) to the ranking model's score for words belonging to the 'best' sense cluster. The success of the sense promotion method depends on two things: having a set of sense clusters that accurately capture the senses of the target word, and having a decently reliable word sense disambiguation (WSD) model to predict the best sense cluster.

Sentence	*In part, prices reflect development of a* ***market*** *structure based on such variables as the number of prints.*
Human-generated Substitutes	business, industry, sector
Model-generated Ranking (P@1/5/10 = 0.0/0.0/0.1)	marketplace, currency, stock, pricing, price, share, economy, listing, transaction, sector, cost, purchasing, workforce, industry, business, trader, circulation, value, buying, supply, segment, enterprise . . .
Best-Fit Sense Cluster	business, competition, economy, industry, institutes, sector
Sense-Promoted Ranking (P@1/5/10 = 0.0/0.6/0.3)	economy, sector, industry, business, competition, institutes, marketplace, currency, stock, pricing, price, share, listing, transaction, cost, purchasing, workforce, trader, circulation, value, buying, supply, segment...

Table 5: In this toy example, the *lexsub* task presents systems with the sentence at top, and requests substitutes for the target word ***market***. Model-generated rankings are compared to human-generated substitutes. In the model-generated ranking, the correct substitutes are scattered throughout the rankings. Using sense clusters as a sense inventory, *sense promotion* predicts the most applicable sense cluster for the target context, and elevates its members to the top of the model's rankings. The resulting sense-promoted rankings have more human-generated substitutes appearing near the top of the list.

3.8.2. Experiments

Sense promotion experiments are run with the dual goals of demonstrating that sense promotion is an effective method for improving the precision of embedding-based lexsub models, and assessing whether the sense clusters generated in this chapter can be applied to this task.

Dataset

For the experiments, target word instances and human-generated substitutes are drawn from the "Concepts in Context" (CoInCo) corpus (Kremer et al., 2014). CoInCo is a lexical substitution dataset containing over 15K sentences corresponding to nearly 4K unique target words. We extract a test set from CoInCo by first finding all target words that have at least

10 sentences, and at least 10 PPDB paraphrases with PPDBSCORE at least 2.0 (to ensure that there are enough PPDB paraphrases of good quality). For each of the 243 resulting targets, we extract a random selection of their corresponding sentences to generate a test set of 2241 sentences in total.

Ranking models

Our approach requires a set of rankings produced by a high-quality lexical substitution model to start. We generate substitution rankings for each target/sentence pair in the test sets using two contemporary models based on word embeddings: ADDCOS (Melamud et al., 2015b), and CONTEXT2VEC (Melamud et al., 2016). In each case, the set of possible substitutes to be ranked for each target word is taken to be all of that target's paraphrases from PPDB-XXL.

The first set of rankings comes from the ADDCOS model of Melamud et al. (2015b). ADDCOS quantifies the fit of substitute word s for target word t in context C by measuring the semantic similarity of the substitute to the target, and the similarity of the substitute to the context:

$$AddCos(s,t,W) = \frac{|W| \cdot cos(s,t) + \sum_{w \in W} cos(s,w)}{2 \cdot |W|} \tag{3.16}$$

The vectors s and t are word embeddings of the substitute and target generated by the *skip-gram with negative sampling* model (Mikolov et al., 2013b,a). The context W is the set of words appearing within a fixed-width window of the target t in a sentence (we use a window (cwin) of 1), and the embeddings c are context embeddings generated by *skip-gram*. In our implementation, we train 300-dimensional word and context embeddings over the 4B words in the Annotated Gigaword (AGiga) corpus (Napoles et al., 2012) using the gensim word2vec package (Řehůřek and Sojka, 2010).[4]

[4]The `word2vec` training parameters we use are a context window of size 3, learning rate *alpha* from 0.025 to 0.0001, minimum word count 100, sampling parameter $1e^{-4}$, 10 negative samples per target word, and 5

The second set of rankings comes from the CONTEXT2VEC model of Melamud et al. (2016). It is more complex than ADDCOS, and has modestly outperformed ADDCOS on the SemEval-2007 lexsub benchmark (McCarthy and Navigli, 2007). Instead of representing the sentential context using the embeddings of neighboring words, it embeds the entire sentence using a bi-directional long short-term memory (LSTM) neural network (Zhou and Xu, 2015; Lample et al., 2016) followed by a multi-layer perceptron. Lexical substitutes are ranked based on the cosine similarity of the substitute's word embedding with the sentence embedding. We train the CONTEXT2VEC model on the Annotated Gigaword corpus using its default settings.

Sense inventories

We assess the performance of sense clusters produced using the best spectral clustering method of Section 3.2.2, which used entailments, $PPDB_{2.0}Score$ similarities, and $sim_{DISTRIB}$ to choose k SPECTRAL.

For each of the 243 target words in the CoInCo test set, we extract paraphrases from PPDB-XXL having PPDBSCORE over thresholds of 2.0 and 2.3. We then cluster each paraphrase set using the spectral method. This results in two 'sense inventories' for evaluation: SPECTRAL:PPDBSCORE $\geq$2.0, and SPECTRAL:PPDBSCORE $\geq$2.3. We also generate the set of extended WordNet synsets (WORDNET+) for each target as a baseline. Recall from the earlier experiments that extended WordNet synsets are composed of lemmas for each of the target word's WordNet synsets, plus their direct hypernyms and direct hyponyms.

Performing word sense disambiguation

Three methods are compared for selecting the best sense cluster given a target word instance in context.

The first *Oracle* method provides an upper bound on the sense promotion performance

training epochs.

of each sense inventory. In this setting, we assume that there exists a WSD oracle which chooses the cluster that maximizes the sum of precision-at-x scores for $x \in \{1, 3, 5, 10\}$.

The second *Random* method provides a lower bound. Here we run five iterations of choosing a random sense cluster for promotion, and calculate the average sense-filtered GAP score over the five iterations.

The third *BestFit* method uses a simple WSD method to predict the correct sense. For a target word in context, we first generate the AddCos score for all words appearing in the sense inventory. We then multiply each word's AddCos score by its ppdbscore with the target word, and take the set of top-5 scoring words. We then choose as the 'best-fit' the cluster with greatest overlap with the top-5 set. This 'best-fit' method finds the sense that aligns with the top-ranked substitutes, and contains words with a strong paraphrase relationship with the target.[5]

Results

We first generate *Original* AddCos and Context2Vec model rankings for each of the roughly 2k instances in the CoInCo test set, and report the average *Original* precision metrics (P@{1/3/5}). Then, for each experimental combination of ranking model, sense inventory, and WSD method, we perform sense promotion over each model's rankings and report the average sense-promoted precision scores. Results are given in Table 6.

While the AddCos lexsub model out-performs the Context2Vec model when used on its own in terms of original rankings, we see that for both lexsub models, running sense promotion using any of the three sense inventories with Oracle WSD indicates that there is potential to increase the precision of the top-1/3/5 ranked substitutes substantially. Furthermore, using the simple BestFit WSD method, while not reaching the upper bounds of precision implied by the Oracle experiment, leads to significant improvements for all sense

[5]The ppdbscore itself was shown to be a strong method for ranking substitute paraphrases in context by Apidianaki (2016).

Context2Vec				
Sense Inv.	Original	Random	BestFit	Oracle
spectral:ppdbscore ≥2.0	14.5 / 12.1 / 11.1	15.7 / 13.0 / 11.8	19.1 / 16.3 / 14.9	35.7 / 26.2 / 22.1
spectral:ppdbscore ≥2.3		19.6 / 16.8 / 15.6	23.2 / 20.5 / 18.5	37.1 / 28.2 / 23.8
WordNet+		22.9 / 17.3 / 14.7	31.7 / 24.2 / 20.1	63.2 / 38.8 / 28.6
AddCos				
Sense Inv.	Original	Random	BestFit	Oracle
spectral:ppdbscore ≥2.0	28.3 / 21.5 / 18.2	22.7 / 17.3 / 14.9	30.4 / 24.2 / 20.1	48.3 / 33.0 / 26.3
spectral:ppdbscore ≥2.3		27.6 / 21.0 / 18.6	31.7 / 25.9 / 21.7	48.1 / 33.7 / 26.9
WordNet+		25.2 / 20.4 / 18.6	34.9 / 26.9 / 22.6	66.0 / 42.0 / 32.1

Table 6: Average P@{1/3/5} scores achieved by lexsub models context2vec and AddCos before (*Original*) and after (*Random*, *BestFit*, and *Oracle*) sense cluster promotion.

inventories as well. For example, by running sense promotion over AddCos rankings using BestFit WSD with the spectral:ppdbscore ≥2.3 sense clusters, the precision of the top-5 ranked substitutes increases from 18.2% to 21.7% – a nearly 20% relative improvement. This validates that sense promotion is an effective, yet simple, method for improving the precision of embedding-based lexsub models using sense clusters.

In general, using the smaller, higher-quality spectral:ppdbscore ≥2.3 clusters for sense promotion results in greater precision gains than using the larger spectral:ppdbscore ≥2.0 clusters. However, neither of our automatically-generated sense inventories produce gains as dramatic as those resulting from sense filtering with WordNet+ clusters. This suggests that the hand-crafted WordNet senses better capture sense distinctions than our automatically-generated sense clusters.

We find that the random sense promotion produces no improvement over AddCos rankings, and minimal improvement over the Context2Vec rankings. Promoting using the BestFit WSD method always out-performs random sense promotion.

To give some concrete examples of how sense promotion with the various inventories works, the sense-promoted output for several CoInCo instances is given in Table 7. The examples help to highlight a few relevant points about the sense clustering method. First, it is important to note that sense promotion preserves the relative lexsub model's original ranking of each word within the selected cluster; this is why the correct substitute *sorrowful* in the second example is still ranked in fifth place under Oracle filtering for the SPECTRAL:PPDBSCORE $\geq$2.3 sense clusters, after several proposed substitutes that are not in the gold set. This also explains how the Oracle P@1 score for the larger SPECTRAL:PPDBSCORE $\geq$2.0 sense clusters can be higher than that for the smaller SPECTRAL:PPDBSCORE $\geq$2.3 clusters (37.1 vs 35.7). Second, recall that the WordNet+ sense inventory can assign one word to multiple sense clusters. In the second example, the word *sad* appears in two WordNet+ sense clusters, and thus is in the selected cluster under both the Oracle and BestFit methods.

Sentence	*In a blink of the strobe light, he was on his feet and dashing from the* ***room****.*
Gold Subs	chamber; area; space; quarter; place
Top-5 AddCos Original	door; bathroom; table; lavatory; ballroom
Top-5 SPECTRAL:PPDBSCORE $\geq$2.3 Oracle	compartment; area; desk; space; headroom
Top-5 3SPECTRAL:PPDBSCORE $\geq$2.3 BestFit	ballroom; classroom; lounge; courtroom; bedroom
Top-5 WordNet+ Oracle	door; bathroom; lavatory; ballroom; classroom
Top-5 WordNet+ BestFit	door; bathroom; lavatory; ballroom; classroom
Sentence	*I'm* ***sorry*** *things happened this way.*
Gold Subs	sad; regretful; sorrowful; angry; unhappy
Top-5 AddCos Original	sad; happy; ashamed; regretful; afraid
Top-5 SPECTRAL:PPDBSCORE $\geq$2.3 Oracle	regretful; afraid; disappointed; unfortunate; sorrowful
Top-5 SPECTRAL:PPDBSCORE $\geq$2.3 BestFit	regretful; afraid; disappointed; unfortunate; sorrowful
Top-5 WordNet+ Oracle	regretful; bad; sad; happy; ashamed
Top-5 WordNet+ BestFit	sad; distressing; pitiful; deplorable; lamentable

Table 7: Examples of sense promotion output. Human-annotated substitutes are shown in blue.

3.9. Conclusion

This chapter has examined how paraphrases can be applied to the task of learning the possible senses, or meanings, of a target word. Bilingually-induced paraphrases from PPDB played a central role in two ways. First, based on the assumption that a target word's paraphrase set contains terms pertaining to each of its senses, we clustered the paraphrases within that set to discriminate the target word's different senses. Second, we showed that using a paraphrase-based signal (PPDBSCORE) to measure the similarity between terms to be clustered is an effective way to ensure each cluster contains terms that share a common meaning.

In our experiments, we experimented with two clustering algorithms (Spectral and HGFC) and five similarity metrics for paraphrase sense clustering. The results indicate that the PPDBSCORE similarity metric consistently produces high-quality clusters when evaluated against either WordNet synsets or a crowd-sourced dataset of ground-truth sense clusters, regardless of the clustering algorithm used. However, our overall best scores were produced by combining the PPDBSCORE metric for measuring term similarity with a monolingual distributional similarity metric for selecting the optimal number of sense clusters, showing that the two types of features are complementary. When evaluated against WordNet synsets, the sense clusters produced by the best Spectral Clustering algorithm give a 64% relative improvement in paired F-Score over the closest baseline.

The second half of this chapter focused on applying the automatically-induced sense clusters to the downstream task of lexical substitution. Most recent lexical substitution models, like AddCos and Context2Vec, use word and context embeddings to propose appropriate ranked substitutes for a target word in context that are both similar in meaning to the original target word, and a good fit for the particular context. These models do not explicitly model word sense. We proposed a simple post-processing method, called 'sense promotion,' that uses sense clusters to improve the precision of embedding-based lexical substitution models by boosting the rank of substitutes that belong to the most appropriate sense cluster given the context. Applying sense promotion with a set of PPDB sense clusters generated using our spectral method led to a 12% improvement in average precision-at-1 and a 19% improvement in average precision-at-5 of AddCos lexical substitution rankings over a dataset of roughly 2000 target word instances (Kremer et al., 2014).

CHAPTER 4 : Learning Scalar Adjective Intensity from Paraphrases

4.1. Introduction

The previous chapter proposed a method for using signals from bilingually-induced paraphrases to discriminate word sense. In this chapter we examine the use of paraphrase-based signals to another task in lexical semantics: predicting the relative intensity between two scalar adjectives.

Semantically similar adjectives are not fully interchangeable in context. Although *hot* and *scalding* are related, the statement *"the coffee was hot"* does not imply the coffee was *scalding*. *Hot* and *scalding* are scalar adjectives that describe *temperature*, but they are not interchangeable because they vary in intensity. A native English speaker knows that their relative intensities are given by the ranking *hot* < *scalding*. Understanding this distinction is important for language understanding tasks such as sentiment analysis (Pang and Lee, 2008), question answering (de Marneffe et al., 2010), and textual inference (Dagan et al., 2006).

particularly pleased	$\leftrightarrow$	*ecstatic*
quite limited	$\leftrightarrow$	*restricted*
rather odd	$\leftrightarrow$	*crazy*
so silly	$\leftrightarrow$	*dumb*
completely mad	$\leftrightarrow$	*crazy*

Figure 18: Examples of paraphrases from PPDB of the form $RB\ JJ_u \leftrightarrow JJ_v$ which can be used to infer pairwise intensity relationships ($JJ_u < JJ_v$).

Existing lexical resources such as WordNet (Miller, 1995; Fellbaum, 1998) do not include the relative intensities of adjectives. As a result, there have been efforts to automate the process of learning intensity relations, as discussed earlier in Section 2.3.2 (e.g. Sheinman and Tokunaga (2009), de Melo and Bansal (2013), Wilkinson (2017), etc.). Many existing approaches rely on *pattern-based* or *lexicon-based* methods to predict the intensity ranking of adjectives. Pattern-based approaches search large corpora for lexical patterns that indicate an intensity relationship – for example, *"not just X, but Y"* implies $X < Y$ (see Table 9 for other examples). As with pattern-based approaches for other tasks (such as hypernym discovery (Hearst, 1992)), they are precise but have relatively sparse coverage of comparable adjectives, even when using web-scale corpora (de Melo and Bansal, 2013; Ruppenhofer et al., 2014). Lexicon-based approaches employ resources that map an adjective to a real-valued number that encodes both intensity and polarity (e.g. *good* might map to 1 and *phenomenal* to 5, while *bad* maps to -1 and *awful* to -3). They can also be precise, but may not cover all adjectives of interest. Examples from the lexicon of Taboada et al. (2011), used in this study, are given in Table 8

Adjective	Score
exquisite	5
beautiful	4
appealing	3
above-average	2
okay	1
ho-hum	-1
pedestrian	-2
gross	-3
grisly	-4
abhorrent	-5

Table 8: Examples of scores for scalar adjectives describing appearance from the SO-CAL lexicon (Taboada et al., 2011). Score magnitude indicates intensity.

Weak-Strong Patterns	Strong-Weak Patterns
* (,) but not *	not * (,) just *
* (,) if not *	not * (,) still *
not only * but *	not * (,) though still *
* (,) (and/or) almost *	* (,) or very *

Table 9: Examples of adjective ranking patterns used in de Melo and Bansal (2013).

We propose paraphrases as a new source of evidence for the relative intensity of scalar adjectives. Specifically, adjectival paraphrases, such as *really great* $\leftrightarrow$ *phenomenal*, can be exploited to uncover intensity relationships. A paraphrase pair of the above form, where one phrase is composed of an intensifying adverb and an adjective (*really great*) and the other is a single-word adjective (*phenomenal*), provides evidence that *great* $<$ *phenomenal*. By drawing this evidence from large, automatically-generated paraphrase resources like PPDB, it is possible to obtain high-coverage pairwise adjective intensity predictions at reasonably high accuracy.

We demonstrate the usefulness of paraphrase evidence for inferring relative adjective intensity in two tasks: ordering sets of adjectives along an intensity scale, and inferring the polarity of indirect answers to *yes/no* questions. In both cases, we find that combining the relatively noisy, but high-coverage, paraphrase evidence with more precise but low-coverage pattern- or lexicon-based evidence improves overall quality.

Relative intensity is just one of several dimensions of gradable adjective semantics. In addition to intensity scales, a comprehensive model of scalar adjective semantics might also incorporate notions of intensity range (Morzycki, 2015), adjective class (Kamp and Partee, 1995), and scale membership according to meaning (Hatzivassiloglou and McKeown, 1993). In this chapter we take the position that relative intensity is worth studying on its own because it is an important component of adjective semantics, usable directly for some NLP tasks such as sentiment analysis (Pang and Lee, 2008), and as part of a more comprehensive model for other tasks like question answering (de Marneffe et al., 2010).

4.2. Paraphrase-based Intensity Evidence

Adjectival paraphrases provide evidence about the relative intensity of adjectives. We claim that a paraphrase of the form $RB\ JJ_u \leftrightarrow JJ_v$ – where one phrase is comprised of an adjective modified by an intensifying adverb ($RB\ JJ_u$), and the other is a single-word adjective (JJ_v) – is evidence that the first adjective is less intense than the second ($JJ_u < JJ_v$). Here, we propose a new method for encoding this evidence and using it to make pairwise adjective intensity predictions. First, a graph (JJGraph) is formed to represent over 36k adjectival paraphrases having the specified form (Figure 19). Next, data in the graph are used to make pairwise adjective intensity predictions.

4.2.1. Identifying Informative Adjectival Paraphrases

In JJGraph, nodes are adjectives, and each directed edge ($\mathrm{JJ}_u \xrightarrow[\mathrm{RB}]{} \mathrm{JJ}_v$) corresponds to an adjectival paraphrase of the form $RB\ JJ_u \leftrightarrow JJ_v$ – for example, *very tall* $\leftrightarrow$ *large* – where one 'phrase' (JJ_v) is an adjective and the other ($\mathrm{RB\ JJ}_u$) is an adjectival phrase containing

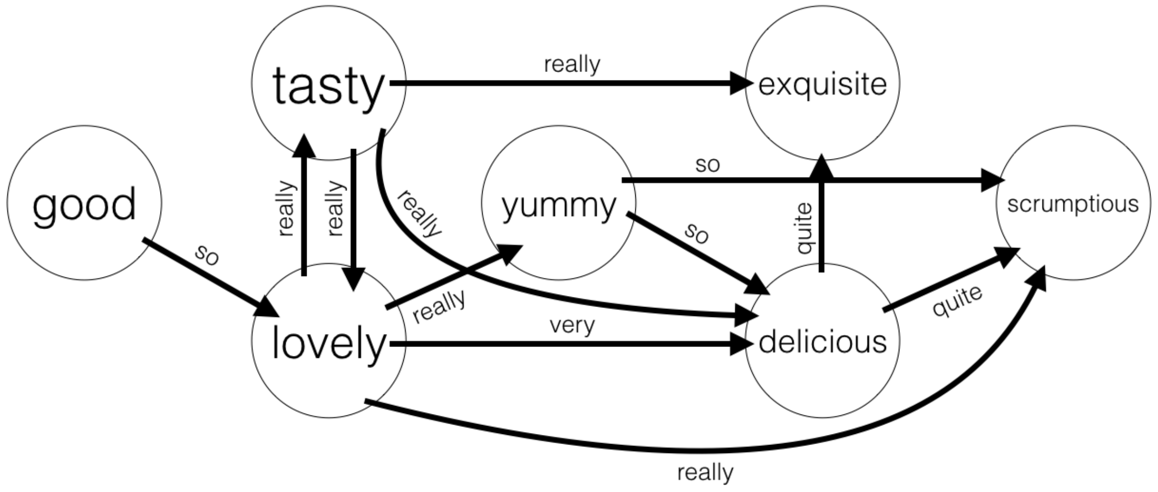


Figure 19: A subgraph of JJGraph, depicting its directed graph structure.

an adverb and adjective (see Figure 18 for examples).

The first step in creating JJGraph is to identify adjectival phrase (ADJP) paraphrases in PPDB-XXL that match the specified template $RB\ JJ_u \leftrightarrow JJ_v$. We search for such paraphrases as follows.

Given an ADJP paraphrase pair, we denote as P_1 the phrase with longer token length, and P_2 the shorter phrase. We assume that P_2 consists of a single adjective, and P_1 consists of an adjective modified by an adverb. More specifically, within P_1 of length n, we identify the adjective as the last token, and the adverbial modifier the concatenated tokens from the first to $(n-1)^{th}$ token. For the purposes of this study, phrases where the adverb meets one of the following criteria are ignored: longer than 4 tokens; consists of a single character; consists of the word *not*; ends with one of the tokens *about*, *and*, *in*, *or*, *the*, or *to*; or contains digits.

4.2.2. Identifying Intensifying Adverbs

Adverbs in PPDB can be intensifying or de-intensifying. An *intensifying* adverb (e.g. *very*, *totally*) strengthens the adjectives it modifies. In contrast, a *de-intensifying* adverb (e.g. *slightly*, *somewhat*) weakens the adjectives it modifies. Since edges in JJGraph ideally point

Round 1				
	very	hard	$\leftrightarrow$	harder
	kinda	hard	$\leftrightarrow$	harder
	so	hard	$\leftrightarrow$	harder
	pretty	hard	$\leftrightarrow$	harder

$\Downarrow$

Round 2				
	very	*pleasant*	$\leftrightarrow$	*delightful*
	kinda	*hard*	$\leftrightarrow$	*tricky*
	so	*wonderful*	$\leftrightarrow$	*brilliant*
	pretty	*simple*	$\leftrightarrow$	*plain*

$\Downarrow$

Round 3				
	more	pleasant	$\leftrightarrow$	delightful
	really	hard	$\leftrightarrow$	tricky
	truly	wonderful	$\leftrightarrow$	brilliant
	quite	simple	$\leftrightarrow$	plain

Figure 20: Bootstrapping process for identifying intensifying adverbs. The adverbs found in Rounds 1 and 3 are used to build intensifying edges in JJGraph.

in the direction of increasing intensity, the first step in the process of creating JJGraph is to identify a set of adverbs that are likely intensifiers to be included as edges.

For this purpose, we generate a set R of likely intensifying adverbs within PPDB using a bootstrapping approach (Figure 20). The process starts with a small seed set of adjective pairs having a known intensity relationship. The seeds are pairs (j_u, j_v) from PPDB-XXL[1] such that j_u is a base-form adjective (e.g. *hard*), and j_v is its **comparative** or **superlative** form (e.g. *harder* or *hardest*)[2]. Using the seeds, we identify intensifying adverbs by finding adjectival paraphrases in PPDB of the form $(r_i j_u \leftrightarrow j_v)$; because $j_u < j_v$, adverb r_i is inferred to be intensifying (Round 1). All such r_i are added to initial adverb set R_1. The process continues by extracting paraphrases $(r_i j_{u'} \leftrightarrow j_{v'})$ with $r_i \in R_1$, indicating additional adjective pairs $(j_{u'}, j_{v'})$ with intensity direction inferred by r_i (Round 2). For example, if $r_i =$ very is an adverb identified in Round 1, then finding the paraphrase *very pleasant* $\leftrightarrow$ *delightful* in Round 2 would lead us to infer that *pleasant* $<$ *delightful.* Finally, the adjective pairs extracted in this second iteration are used to identify additional intensifying adverbs R_3, which are added to the final set $R = R_1 \cup R_3$ (Round 3). To continue with the previous

[1]PPDB comes in six increasingly large sizes from S to XXXL; larger collections have wider coverage but lower precision. Our work uses XXL.

[2]Such pairs were identified by lemmatizing with NLTK's `WordNetLemmatizer` (Loper and Bird (2002)).

example, if the relation *pleasant* $<$ *delightful* is assumed in Round 2, and *super pleasant* $\leftrightarrow$ *delightful* is a paraphrase in PPDB, then *super* will be added to the set of intensifying adverbs R_3.

In all, this process generates a set of 610 adverbs. Examination of the set shows that the process does capture many intensifying adverbs like *very* and *abundantly*, and excludes many de-intensifying adverbs appearing in PPDB like *far less* and *not as*. However, due to the noise inherent in PPDB itself and in the bootstrapping process, there are also a few de-intensifying adverbs included in R (e.g. *hardly*, *kind of*) as well as adverbs that are neither intensifying nor de-intensifying (e.g. *ecologically*). It will be important to take this noise into consideration when using JJGraph to make pairwise intensity predictions.

4.2.3. Building JJGraph

JJGraph is built by extracting all 36,756 adjectival paraphrases in PPDB of the specified form $RB\ JJ_u \leftrightarrow JJ_v$, where the adverb belongs to R. The resulting graph has 3,704 unique adjective nodes. JJGraph is a multigraph, as there are frequently multiple intensifying relationships between pairs of adjectives. For example, the paraphrases *pretty hard* $\leftrightarrow$ *tricky* and *really hard* $\leftrightarrow$ *tricky* are both present in PPDB. There can also be contradictory or cyclic edges in JJGraph, as in the example depicted in the JJGraph subgraph in Figure 19, where the adverb *really* connects *tasty* to *lovely* and vice versa. Self-edges are allowed (e.g. *really hard* $\leftrightarrow$ *hard*).

4.2.4. Pairwise Intensity Prediction

Examining the directed adverb edges between two adjectives j_u and j_v in JJGraph provides evidence about the relative intensity relationship between them. However, it has just been noted that JJGraph is noisy, containing both contradictory/cyclic edges and adverbs that are not uniformly intensifying. Rather than try to eliminate cycles, or manually annotate each adverb with a weight corresponding to its intensity and polarity (Ruppenhofer et al., 2015; Taboada et al., 2011), we aim to learn these weights automatically in the process of predicting pairwise intensity.

Given adjective pair (j_u, j_v), we build a classifier that outputs a score from 0 to 1 indicating the predicted likelihood that $j_u < j_v$. Its binary features correspond to adverb edges from j_u to j_v and from j_v to j_u in JJGraph. The feature space includes only adverbs from R that appear at least 10 times in JJGraph, resulting in features for $m = 259$ unique adverbs in each direction (i.e. from j_u to j_v and vice versa) for $2m = 518$ binary features total. Note that while all adverb features correspond to predicted intensifiers from R, there are some features that are actually de-intensifying due to the noise inherent in the bootstrapping process (Section 4.2.2).

We train the classifier on all 36.7k edges in JJGraph, based on a simplifying assumption that all adverbs in R are indeed intensifiers. For each adjective pair (j_u, j_v) with one or more direct edges from j_u to j_v, a positive training instance for pair (j_u, j_v) and a negative training instance for pair (j_v, j_u) are added to the training set. A logistic regression classifier is trained on the data, using elastic net regularization and 10-fold cross validation to tune parameters.

The model parameters output by the training process are in a feature weights vector $w \in \mathbb{R}^{2m}$ (with no bias term) which can be used to generate a paraphrase-based score for each adjective pair:

$$score_{\text{pp}}(j_u, j_v) = \frac{1}{1 + \exp^{-wx_{uv}}} - 0.5 \tag{4.1}$$

where x_{uv} is the binary feature vector for adjective pair (j_u, j_v). The decision boundary 0.5 is subtracted from the sigmoid activation function so that pairs predicted to have the directed relation $j_u < j_v$ will have a positive score, and those predicted to have the opposite directional relation will have a negative score.

4.2.5. Examining Adverb Weights

One interesting artefact of the adjective pair intensity classifier is the feature weights vector, w, which assigns two numeric weights to each adverb represented in the feature space (one weight corresponding an edge in each direction). Intuitively, the weights might be expected to correspond to the intensification strength of each adverb.

Similarly to adjectives, adverbs can be classified according to their strength or as modifiers. Some studies group adverbs into discrete categories such as maximizers (*absolutely*, *completely*, *totally*, etc), boosters (*very*, *terribly*, *highly*, etc), moderators (*rather*, *pretty*, *fairly*), or diminishers (*a little*, *slightly*, *somewhat*) (Paradis, 1997) – with each category being weaker than the previous. Ruppenhofer et al. (2015) took a slightly different approach, asking crowd workers to assign a score to each of 14 adverbs according to their place along an intensity scale. In order to examine whether the adverb feature weights from our adjective intensity classifier are reflective of their strength as modifiers, we compare Ruppenhofer's adverb scores to the mean weight of each adverb in our feature space:

$$\text{weight}(r) = \frac{w_{r:uv} - w_{r:vu}}{2} \tag{4.2}$$

where $w_{r:uv}$ gives the feature weight of adverb r in the $j_u \rightarrow j_v$ direction, and $w_{r:vu}$ gives the feature weight of adverb r in the $j_v \rightarrow j_u$ direction. If an adverb has high weight, this means that it is strongly indicative of a weak-strong relationship for adjective pair (j_u, j_v) when it modifies j_u, and strongly indicative of a strong-weak relationship when it modifies j_v. The comparison between the mean feature weight and the Ruppenhofer et al. (2015) scores is depicted in Figure 21.

Based on this limited analysis, it does not appear that the classifier weights correlate well with adverb intensity. In particular, adverbs *slightly* and *almost* have a much higher than expected classifier weight, given their status as diminishers. This may be the result of the

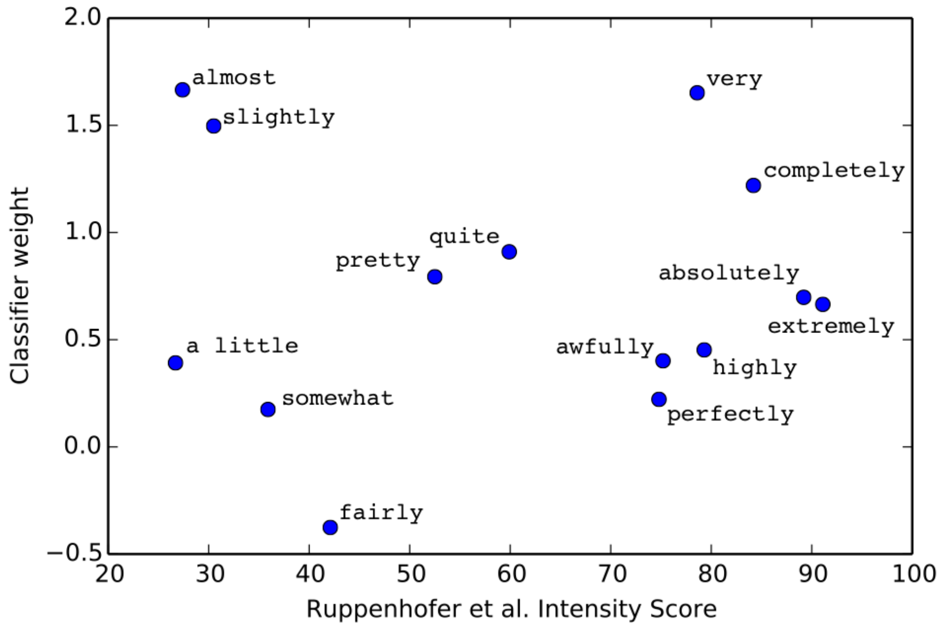


Figure 21: A comparison between human-annotated adverb intensity weights from Ruppenhofer et al. (2015), and mean adverb weights from the pairwise intensity classifier (Eq. 4.2).

simplifying assumption that was made when training the classifier, namely that all adverbs in the graph were intensifiers.

4.3. Other Intensity Evidence

Our experiments compare the proposed paraphrase approach with existing pattern- and lexicon-based approaches.

4.3.1. Pattern-based Evidence

We experiment with the pattern-based approach of de Melo and Bansal (2013). Given a pair of adjectives to be ranked by their intensity, de Melo and Bansal (2013) cull intensity patterns from Google n-Grams (Thorsten and Franz, 2006) as evidence of their intensity order. Specifically, they identify 8 types of *weak-strong* patterns (e.g. *"X, but not Y"*) and 7 types of *strong-weak* patterns (e.g. *"not X, but still Y"*) that are used as evidence about the directionality of the intensity relationship between adjectives. Given an adjective pair (j_u, j_v), an overall pattern-based *weak-strong* score is calculated:

$$score_{\text{pat}}(j_u, j_v) = \frac{(W_u - S_u) - (W_v - S_v)}{\text{count}(j_u) \cdot \text{count}(j_v)} \tag{4.3}$$

where W_u and S_u quantify the pattern evidence for the *weak-strong* and *strong-weak* intensity relations respectively for the pair (j_u, j_v), and W_v and S_v quantify the pattern evidence for the pair (j_v, j_u). W_u and S_u are calculated as:

$$\begin{aligned} W_u &= \frac{1}{P_1} \sum_{p_1 \in P_{ws}} \text{count}(p_1(j_u, j_v)) \\ S_u &= \frac{1}{P_2} \sum_{p_2 \in P_{sw}} \text{count}(p_2(j_u, j_v)) \end{aligned} \tag{4.4}$$

W_v and S_v are calculated similarly by swapping the positions of j_u and j_v. For example, given pair (*good*, *great*), W_u might incorporate evidence from patterns *"good, but not great"* and *"not only good but great"*, while S_v might incorporate evidence from the pattern *"not great, just good"*. P_{ws} denotes the set of *weak-strong* patterns, P_{sw} denotes the set of *strong-weak* patterns, and P_1 and P_2 give the total counts of all occurrences of any pattern in P_{ws} and P_{sw} respectively. The score is normalized by the frequencies of j_u and j_v in order to avoid bias due to high-frequency adjectives. As with the paraphrase-based scoring mechanism (Equation 4.1), scores output by this method can be positive or negative, with positive scores being indicative of a *weak-strong* relationship from j_u to j_v. Note that $score(j_u, j_v) = -score(j_v, j_u)$.

4.3.2. Lexicon-based Evidence

We use the manually-compiled SO-CAL[3] lexicon as our third, lexicon-based method for inferring intensity. The SO-CAL lexicon assigns an integer weight in the range $[-5, 5]$ to 2,826 adjectives. The sign of the weight encodes sentiment polarity (positive or negative), and the value encodes intensity (e.g. *atrocious*, with a weight of -5, is more intense than *unlikable*, with a weight of -3). SO-CAL is used to derive a pairwise intensity prediction for adjectives (j_u,j_v) as follows:

$$\begin{aligned} score_{\mathrm{socal}}(j_u, j_v) = |L(j_v)| - |L(j_u)|, \\ \text{iffsign}(j_u) = \text{sign}(j_v) \end{aligned} \tag{4.5}$$

where $L(j_v)$ gives the lexicon weight for j_v. Note that $score_{socal}$ is computed only for adjectives having the same polarity direction in the lexicon; otherwise the score is undefined. This is because adjectives belonging to different half scales, such as *freezing* and *steaming*, are frequently incomparable in terms of intensity (de Marneffe et al., 2010).

4.3.3. Combining Evidence

While the pattern-based and lexicon-based pairwise intensity scores are known to be precise but low-coverage (de Melo and Bansal, 2013; Ruppenhofer et al., 2015), we expect that the paraphrase-based score will produce higher coverage at lower accuracy. Thus we also experiment with scoring methods that combine two or three score types. When combining two metrics $\mathbf{x}$ and $\mathbf{y}$ to generate a score for a pair (j_u, j_v), we simply use the first metric $\mathbf{x}$ if it can be reliably calculated for the pair, and back off to metric $\mathbf{y}$ otherwise. More formally, the combined score for metrics $\mathbf{x}$ and $\mathbf{y}$ is given by:

[3]`https://github.com/sfu-discourse-lab/SO-CAL`

$$\begin{aligned} \mathit{score}_{\text{x+y}}(j_u, j_v) = \alpha_{\text{x}} \cdot g_{\text{x}}(\mathit{score}_{\text{x}}(j_u, j_v)) \\ + (1 - \alpha_{\text{x}}) \cdot g_{\text{y}}(\mathit{score}_{\text{y}}(j_u, j_v)) \end{aligned} \tag{4.6}$$

where $\alpha_{\text{x}} \in \{0, 1\}$ is a binary indicator corresponding to the condition that $\mathit{score}_{\text{x}}$ can be reliably calculated for the adjective pair, and $g_{\text{x}}(\cdot)$ is a scaling function (see below). If $\alpha_{\text{x}} = 1$, then $\mathit{score}_{\text{x}}$ is used. Otherwise, if $\alpha_{\text{x}} = 0$, then we default to $\mathit{score}_{\text{y}}$. When combining three metrics x, y, and z, the combined score is given by:

$$\begin{aligned} \mathit{score}_{\text{x+y+z}}(j_u, j_v) = \alpha_{\text{x}} \cdot g_{\text{x}}(\mathit{score}_{\text{x}}(j_u, j_v)) \\ + (1 - \alpha_{\text{x}}) \cdot \mathit{score}_{\text{y+z}}(j_u, j_v) \end{aligned} \tag{4.7}$$

The criteria for having $\alpha_{\text{x}} = 1$ varies depending on the metric type. For pattern-based evidence (**x**='pat'), $\alpha_{\text{x}} = 1$ when adjectives j_u and j_v appear together in any of the intensity patterns culled from Google *n*-grams (e.g. a pattern like "j_u, but not j_v" exists). For lexicon-based evidence (**x**='socal'), $\alpha_{\text{x}} = 1$ when both j_u and j_v are in the SO-CAL vocabulary, and have the same polarity (i.e. are both positive or both negative). For paraphrase-based evidence (**x**='pp'), $\alpha_{\text{x}} = 1$ when j_u and j_v have one or more edges directly connecting them in JJGraph.

Since the metrics to be combined may have different ranges, we use a scaling function $g_{\text{x}}(\cdot)$ to make the scores output by each metric directly comparable:

$$g_{\text{x}}(w) = \text{sign}(w) \cdot \left(\frac{\log(|w|) - \mu_{\text{x}}}{\sigma_{\text{x}}} + \gamma \right) \tag{4.8}$$

where μ_x and σ_x are the estimated population mean and standard deviation of $\log(score_\text{x})$ (estimated over all adjective pairs in the dataset), and γ is an offset that ensures positive scores remain positive, and negative scores remain negative. In our experiments we set $\gamma = 5$.

By way of comparison, Table 10 shows the pairwise intensity predictions made by each of the individual score types, and a combination of the three, on eight randomly selected adjective pairs from the datasets used in Section 4.4.

Pair	*gold*	$score_\text{pp}$	$score_\text{pat}$	$score_\text{socal}$	$score_\text{pat+socal+pp}$
gigantic, huge	$>$	$>$	$--$	$--$	$>$
good, great	$<$	$<$	$<$	$<$	$<$
huge, enormous	$<$	$<$	$--$	$<$	$<$
gigantic, big	$>$	$>$	$--$	$--$	$>$
few, many	$<$	$--$	$<$	$--$	$<$
light, bright	$<$	$>$	$>$	$--$	$>$
quick, speedy	$<$	$<$	$--$	$>$	$>$
fresh, new	$<$	$--$	$--$	$--$	$--$

Table 10: Pairwise intensity direction predicted by each of the individual score types, and a combination of the three. The symbol $<$ indicates a weak-strong pair, $>$ indicates a strong-weak pair, and $--$ indicates that the score type could not be computed for that pair.

4.4. Ranking Adjective Sets by Intensity

The first experimental application for the different paraphrase evidence is an existing model for predicting a global intensity ordering within a set of adjectives. Global ranking models are useful for inferring intensity comparisons between adjectives for which there is no explicit evidence. For example, in ranking three adjectives like *warm*, *hot*, and *scalding*, there may be direct evidence indicating *warm* $<$ *hot* and *hot* $<$ *scalding*, but no way of directly comparing *warm* to *scalding*. Global ranking models infer that *warm* $<$ *scalding* based on evidence from the other adjective pairs in the scale.

Dataset	# of Scales	Min/Max/Mean Scale Size	# of Unordered (unequal) Pairs	Example Scale
deMelo	87	3 / 8 / 3.90	524 (466)	$\{\text{clean}\} < \{\text{spotless, immaculate}\}$
Crowd	79	2 / 8 / 3.18	293 (250)	$\{\text{low}\} < \{\text{limited}\} < \{\text{scarce}\}$
Wilkinson	21	2 / 5 / 2.81	61 (61)	$\{\text{dry}\} < \{\text{arid}\} < \{\text{parched}\}$

Table 11: Characteristics of the scalar adjective datasets used for evaluation. The deMelo scale example shows an instance of an equally-intense pair (*spotless*, *immaculate*).

4.4.1. Global Ranking Model

We adopt the mixed-integer linear programming (MILP) approach of de Melo and Bansal (2013) for generating a global intensity ranking. This model takes a set of adjectives $A = \{a_1, \ldots, a_n\}$ and directed, pairwise adjective intensity scores $score(a_i, a_j)$ as input, and assigns each adjective a_i a place along a linear scale $x_i \in [0, 1]$. The adjectives' assigned values define the global ordering. If the predicted weights used as input are inconsistent, containing cycles, the model resolves these by choosing the globally optimal solution.

Recall that all pairwise scoring metrics produce a positive score for adjective pair (j_u, j_v) when it is likely that $j_u < j_v$, and a negative score otherwise. Consequently, the MILP approach should result in $x_u < x_v$ when $score(j_u, j_v)$ is positive, and $x_u > x_v$ otherwise. This goal is achieved by maximizing the objective function:

$$\sum_{u,v} \text{sign}(x_v - x_u) \cdot score(j_u, j_v) \tag{4.9}$$

de Melo and Bansal (2013) propose the following MILP formulation for maximizing this objective, which we implement using the Gurobi ILP software (Gurobi Optimization, 2016) and utilize in our experiments:

$$\begin{aligned}
\max_{u,v} \quad & \sum_{u,v}(w_{uv} - s_{uv}) \cdot score(j_u, j_v) & \\
\text{s.t.} \quad & d_{uv} = x_v - x_u & \forall u, v \in N \\
& d_{uv} - w_{uv}C \leq 0 & \forall u, v \in N \\
& d_{uv} + (1 - w_{uv})C > 0 & \forall u, v \in N \\
& d_{uv} + s_{uv}C \geq 0 & \forall u, v \in N \\
& d_{uv} - (1 - s_{uv})C < 0 & \forall u, v \in N \\
& x_u \in [0, 1] & \forall u \in N \\
& w_{uv} \in \{0, 1\} & \forall u, v \in N \\
& s_{uv} \in \{0, 1\} & \forall u, v \in N
\end{aligned} \tag{4.10}$$

The variable d_{uv} is a difference variable that captures the difference between x_v and x_u. The constant C is an arbitrarily large number that is at least $\sum_{u,v} |score(j_u, j_v)|$. The variables w_{uv} and s_{uv} are binary indicators that correspond to a *weak-strong* or *strong-weak* relationship between j_u and j_v respectively; the objective encourages $w_{uv} = 1$ when $score(j_u, j_v) > 0$, and $s_{uv} = 1$ when $score(j_u, j_v) < 0$. Note that while de Melo and Bansal (2013) also propose an additional term in the objective that incorporates synonymy evidence from WordNet in their ranking method, we do not implement this part of the model.

4.4.2. Experiments

We experiment with using each of the paraphrase-, pattern-, and lexicon-based pairwise scores as input to the global ranking model in isolation. To examine how the scoring methods perform when used in combination, we also test all possible ordered combinations of 2 and 3 scores.

Experiments are run over three distinct test sets (Table 11). Each dataset contains ordered sets of scalar adjectives belonging to the same *scale*. In general, scalar adjectives describing the same attribute can be ordered along a full scale (e.g. *freezing* to *sweltering*), or a half scale (*warm* to *sweltering*); all three test sets group adjectives into half scales. The three datasets are described here, and their characteristics are given in Table 11.

deMelo (de Melo and Bansal, 2013)[4]. 87 adjective sets are extracted from WordNet 'dumbbell' structures (Gross and Miller, 1990), and partitioned into half-scale sets based on their pattern-based evidence in the Google N-Grams corpus (Thorsten and Franz, 2006). Sets are manually annotated for intensity relations ($<$, $>$, and $=$).

Wilkinson (Wilkinson and Oates, 2016). Twelve adjective sets are generated by presenting crowd workers with small seed sets (e.g. *huge, small, microscopic*), and eliciting similar adjectives. Sets are automatically cleaned for consistency, and then annotated for intensity by crowd workers. While the original dataset contains full scales, we manually sub-divide these into 21 half-scales for use in this study. Details on the modification from full- to half-scales are in Appendix A.6.

Crowd. We also crowdsourced a new set of adjective scales with high coverage of the PPDB vocabulary. In a three-step process, we first asked crowd workers whether pairs of adjectives describe the same attribute (e.g. temperature) and therefore should belong along the same scale. Second, sets of same-scale adjectives were refined over multiple rounds. Finally, workers ranked the adjectives in each set by intensity. The final dataset includes 293 adjective pairs along 79 scales.

We measure the agreement between the gold standard ranking of adjectives along each scale and the predicted ranking using three commonly-used metrics:

Pairwise accuracy. For each pair of adjectives along the same scale, we compare the predicted ordering of the pair after global ranking ($<$, $>$, or $=$) to the gold-standard ordering of the pair, and report overall accuracy of the pairwise predictions.

Kendall's tau (τ_b). This metric computes the rank correlation between the predicted

[4]`http://demelo.org/gdm/intensity/`

($r_P(J)$) and gold-standard ($r_G(J)$) ranking permutations of each adjective scale J, incorporating a correction for ties. Values for τ_b range from -1 to 1, with extreme values indicating a perfect negative or positive correlation, and a value of 0 indicating no correlation between predicted and gold rankings. We report τ_b as a weighted average over scales in each dataset, where weights correspond to the number of adjective pairs in each scale.

Spearman's rho (ρ). We report the Spearman's ρ rank correlation coefficient between predicted ($r_P(J)$) and gold-standard ($r_G(J)$) ranking permutations. For each dataset, we calculate this metric just once by treating each adjective in a particular scale as a single data point, and calculating an overall ρ for all adjectives from all scales.

More detail on each evaluation metric is given in Appendix A.1.

4.4.3. Experimental Results

The results of the global ordering experiment, reported in Table 12, are organized as follows: *Score Accuracy* pertains to performance of the scoring methods alone – prior to global ranking – while *Global Ranking Results* pertains to performance of each scoring method as used in the global ranking algorithm. Within *Score Accuracy* there are two metrics. *Coverage* gives the percent of unique same-scale adjective pairs from the test set that can be directly scored using the given method. For $score_{\text{pat}}$, covered pairs are all those that appear together in any recognized pattern; for $score_{\text{pp}}$, covered pairs are those directly connected in JJGraph by one or more direct edges; for $score_{\text{socal}}$, covered pairs are all those for which both adjectives are in the SO-CAL lexicon and the metric is defined. *Pairwise Accuracy* gives the accuracy of the scoring method (before global ranking) on *just the covered pairs*, meaning that the subset of pairs scored by each method varies. Within *Global Ranking Results*, we report pairwise accuracy, weighted average τ_b, and ρ calculated over *all pairs* after ranking – including both pairs that are covered by the scoring method, and those whose pairwise intensity relationship has been inferred by the ranking algorithm.

The results indicate that the pairwise score accuracies (before ranking) for $score_{\text{pat}}$ and

		Score Accuracy (before ranking)		Global Ranking Results			
Test Set	Score Type	Coverage	Pairwise Acc.	Pairwise Acc.	Avg. τ_b	ρ	Example Predicted Scale
	$score_{\text{pat}}$	0.48	**0.844**	0.650	**0.633**	**0.583**	{clean} < {spotless, immaculate}*
deMelo	$score_{\text{pp}}$	0.33	0.458	0.307	0.071	0.090	{immaculate, clean} < {spotless}
	$score_{\text{socal}}$	0.28	0.546	0.246	0.110	0.019	{clean} < {spotless} < {immaculate}
	$score_{\text{pat+socal}}$	0.61	0.757	**0.653**	0.609	0.533	{clean} < {spotless} < {immaculate}
	$score_{\text{pat+socal+pp}}$	**0.70**	0.722	0.644	0.564	0.482	{clean} < {spotless} < {immaculate}
	$score_{\text{pat}}$	0.11	**0.784**	0.321	0.203	0.221	{limited, low, scarce}
Crowd	$score_{\text{pp}}$	0.74	0.676	$0.597^{\dagger\dagger}$	$0.437^{\dagger}$	0.405	{low} < {limited} < {scarce}*
	$score_{\text{socal}}$	0.35	0.757	0.421	0.342	0.293	{limited, low, scarce}
	$score_{\text{socal+pp}}$	0.81	0.687	$0.621^{\dagger\dagger}$	$0.470^{\dagger\dagger}$	0.465	{low} < {limited} < {scarce}*
	$score_{\text{socal+pat+pp}}$	**0.82**	0.694	$\mathbf{0.639}^{\dagger\dagger}$	$\mathbf{0.495}^{\dagger\dagger}$	**0.480**	{low} < {limited} < {scarce}*
	$score_{\text{pat}}$	0.44	0.852	0.475	0.441	0.435	{quick} < {speedy, fast}
Wilkinson	$score_{\text{pp}}$	0.80	0.753	0.639	0.419	0.450	{quick} < {fast} < {speedy}*
	$score_{\text{socal}}$	0.31	**0.895**	0.312	0.317	0.422	{fast} < {speedy} < {quick}
	$score_{\text{pat+pp}}$	**0.89**	0.833	$0.738^{\dagger\dagger}$	0.605	0.564	{quick} < {fast} < {speedy}*
	$score_{\text{pat+socal+pp}}$	**0.89**	0.833	$\mathbf{0.754}^{\dagger\dagger}$	**0.638**	**0.611**	{quick} < {fast} < {speedy}*

††: $p \leq .01$ †: $p \leq .05$

Table 12: Pairwise relation prediction and global ranking results for each score type in isolation, and for the best-scoring combinations of 2 and 3 score types on each dataset. For the global ranking accuracy and average τ_b results, we denote with the † symbol scores for metrics incorporating paraphrase-based evidence that significantly out-perform both $score_{\text{pat}}$ and $score_{\text{socal}}$ under the paired Student's t-test, using the Anderson-Darling test to confirm that scores conform to a normal distribution (Fisher, 1935; Anderson and Darling, 1954; Dror et al., 2018). Example output is also given, with correct rankings starred.

$score_{\text{socal}}$ are higher than those of $score_{\text{pp}}$ for all datasets, but that their coverage is relatively limited. The one exception is the deMelo dataset, where $score_{\text{pat}}$ has high coverage because the dataset was compiled specifically by finding adjective pairs that matched lexical patterns in the corpus. For all datasets, highest coverage is achieved using one of the combined metrics that incorporates paraphrase-based evidence.

Figure 22 examines the trade-off between each score type's coverage and accuracy in more detail. Here are presented the percent of *all* unique adjective pairs from the three datasets (878 pairs total) covered by each score type, plotted against the pairwise accuracy of each score type on the pairs it covers. Points to the upper right have both high coverage and high accuracy. Visualized in this way, it is straightforward to see that combining two or three score types increases the percentage of pairs covered, without sacrificing a substantial

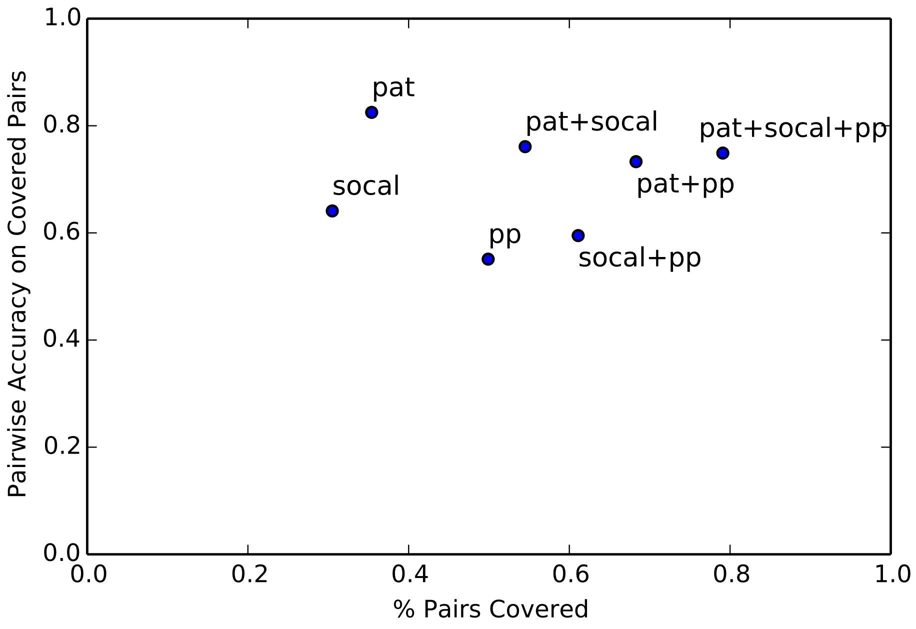


Figure 22: Scatterplot of the proportion of all unique pairs from the 3 datasets (878 pairs total) covered by each score type, versus the pairwise accuracy of each score type on the pairs it covers. Combining the three score types together ($score_{\text{pat+socal+pp}}$) produces the best balance of coverage and accuracy.

amount in terms of accuracy. The best balance of coverage and accuracy is achieved by $score_{\text{pat+socal+pp}}$, which has 75% accuracy at 79% coverage.

The impact of these trends is visible on the Global Ranking Results. When using pairwise intensity scores to compute the global ranking, higher coverage by a metric drives better results, as long as the metric's accuracy is reasonably high. Thus the paraphrase-based $score_{\text{pp}}$, with its high coverage, gets better global ranking results than the other single-method scores for two of the three datasets. Further, we find that boosting coverage with a combined metric that incorporates paraphrase evidence produces the highest post-ranking pairwise accuracy scores overall for all three datasets, and the highest average τ_b and ρ on the Crowd and Wilkinson datasets. We conclude that incorporating paraphrase evidence can improve the quality of this model for ordering adjectives along a scale because it gives high coverage with reasonably high quality.

The performance trends on the deMelo dataset differ from those on the Crowd and Wilkinson datasets. In particular, $score_{\text{pp}}$ and $score_{\text{socal}}$ have substantially lower pre-ranking pairwise accuracy on the pairs they cover in the deMelo dataset than they do for Crowd and Wilkinson: $score_{\text{pp}}$ has an accuracy of just 0.458 on covered pairs in the deMelo dataset, compared with 0.676 and 0.753 on the Crowd and Wilkinson datasets, and score differences for $score_{\text{socal}}$ are similar. The near-random prediction accuracies of $score_{\text{pp}}$ and $score_{\text{socal}}$ on deMelo before ranking lead to near-zero correlation values on this dataset after global ranking. To explore possible reasons for these results, we assessed the level of human agreement with each dataset in terms of pairwise accuracy. For each test set, we asked five crowd workers to classify the intensity direction for each adjective pair (j_u, j_v) in all scales as less than ($<$), greater than ($>$), or equal ($=$). We found that humans agreed with the 'gold standard' direction 65% of the time on the Bansal dataset, versus 70% of the time on the Crowd and Wilkinson datasets. It is possible that the more difficult nature of the Bansal dataset, coupled with its method of compilation (i.e. favoring adjective pairs that co-occur with pre-defined intensity patterns), lead to the lower coverage and lower accuracy of $score_{\text{pp}}$ and $score_{\text{socal}}$ on this dataset.

4.5. Indirect Question Answering

The second task that we address is answering indirect *yes* or *no* questions. Several studies of pragmatics have observed that answers to such polar questions frequently omit an explicit *yes* or *no* response (Grice, 1975; Hirschberg, 1984, 1985; Green and Carberry, 1994, 1999; de Marneffe et al., 2010). For example:

Q: *Did the Eagles win the Super Bowl again?*

A: *They lost the divisional playoff.*

Hirschberg (1985) attributes these indirect responses to attempts by the answering speaker to provide enough information so that the direct response can be derived, while saying as much as she truthfully can that is relevant to the exchange.

In some cases the implied direct answer depends on the relative intensity of adjective modifiers in the question and answer. For example, in the exchange:

Q: *Was he a successful ruler?*

A: *Oh, a tremendous ruler.*

the implied answer is *yes*, which is inferred because *successful* $\leq$ *tremendous* in terms of relative intensity. Conversely, in the exchange:

Q: *Does it have a large impact?*

A: *It has a medium-sized impact.*

the implied answer is *no* because *large* $>$ *medium-sized*.

de Marneffe et al. (2010) compiled an evaluation set for this task by extracting 123 examples of such indirect question-answer pairs (IQAP) from dialogue corpora (including the two examples repeated above). In each exchange, the implied answer (annotated by crowd workers to be *yes* or *no*[5]) depends on the relative intensity relationship between modifiers in the question and answer texts. In their original paper, the authors utilize an automatically-compiled lexicon to make a polarity prediction for each IQAP.

4.5.1. Predicting Answer Polarity

Our goal is to see whether paraphrase-based scores are useful for predicting the polarity of answers in the IQAP dataset. As before, we compare the quality of predictions made using the paraphrase-based evidence with predictions made using pattern-based, lexicon-based, and combined scoring metrics.

To use the pairwise scores for inference, we employ a decision procedure nearly identical to that of de Marneffe et al. (2010). If j_q and j_a are scorable (i.e. have a scorable intensity relationship along the same half-scale), then $j_q \leq j_a$ implies the answer is *yes* (first example above), and $j_q > j_a$ implies the answer is *no* (second example). If the pair of adjectives is

[5]The original dataset contains two additional examples where the answer is annotated as *uncertain*, but de Marneffe et al. (2010) exclude them from the results and so do we.

Given: A dialogue exchange consisting of a polar question and answer, where the answer depends on the relative intensities of distinct modifiers j_q and j_a in the question and answer respectively:

1. if j_q or j_a are missing from the score vocabulary, predict "UNCERTAIN"
2. else, if $score(\mathrm{JJ}_q, \mathrm{JJ}_a)$ is undefined, predict "NO"
3. else, if $score(\mathrm{JJ}_q, \mathrm{JJ}_a) \geq 0$, predict "YES"
4. else, predict "NO"
5. If the question or answer contains negation, map a "YES" answer to "NO and a "NO" answer to "YES"

Figure 23: Decision procedure for using pairwise intensity scores for predicting polarity of an IQAP instance, based on de Marneffe et al. (2010).

not scorable, then the predicted answer is *no*, as the pair could be antonyms or completely unrelated. If either j_q or j_a is missing from the scoring vocabulary, the adjectives are impossible to compare and therefore the prediction is *uncertain*. The full decision procedure is given in Figure 23.

4.5.2. Experiments

The decision procedure in Figure 23 is carried out for the 123 IQAP instances in the dataset, varying the score type. We report the accuracy, and macro-averaged precision, recall, and F1-score of the 85 *yes* and 38 *no* instances, in Table 13 alongside the percent of instances with adjectives out of vocabulary. Only the combined scores for the two best-scoring combinations, $score_{\text{socal+pp}}$ and $score_{\text{socal+pat+pp}}$, are reported.

The simplest baseline of predicting all answers to be "YES" gets highest accuracy in this imbalanced test set, but all score types perform better than the all-"YES" baseline in terms of precision and F1-score. Bouyed by its high precision, the $score_{\text{socal}}$ – which is derived from a manually-compiled lexicon – scored higher than $score_{\text{pp}}$ and $score_{\text{pat}}$. But it mis-predicted 33% of pairs as *uncertain* because of its limited overlap with the IQAP vocabulary. Meanwhile, $score_{pp}$ had relatively high coverage and a mid-level F-score, while $score_{pat}$ scored poorly on this dataset due to its sparsity; while all modifiers in the IQAP dataset

Method	%OOV	Acc.	P	R	F
all-“YES”	.00	**.691**	.346	.500	.409
deMarneffe (2010)	.02	.610	.597	.594	.596
$score_{\text{socal}}$	.33	.504	**.710**	.481	.574
$score_{\text{pp}}$	.09	.496	.568	.533	.550
$score_{\text{pat}}$	.07	.407	.524	.491	.507
$score_{\text{socal+pp}}$	.09	.634	.690	.663	.676
$score_{\text{socal+pat+pp}}$	.06	.642	.684	.683	.684
$score_{\text{socal+pat+pp+demarneffe}}$	.01	.667	.675	**.701**	**.688**

Table 13: Accuracy and macro-averaged precision (P), recall (R), and F1-score (F) over *yes* and *no* responses on 125 question-answer pairs. The percent of pairs having one or both adjectives out of the score vocabulary (and therefore resulting in an *uncertain* prediction) is listed as %OOV.

are in the Google N-grams vocabulary, most do not have observed patterns and therefore return predictions of “NO” (item 2 in Figure 23). As in the global ranking experiments, the paraphrase-based evidence is complementary to the lexicon-based evidence, and thus the combined $score_{\text{socal+pp}}$ and $score_{\text{socal+pat+pp}}$ produce significantly better accuracy than any score in isolation (McNemar’s test, $p < .01$), and also out-perform the original expected ranking method of de Marneffe et al. (2010) (although they do not beat the best-reported score on this dataset, F-score=0.706 (Kim and de Marneffe, 2013)). Further, because the deMarneffe method has high coverage (only 2% of pairs OOV), we can add it as a fourth scoring type to the combined $score_{\text{socal+pat+pp}}$ in order to increase its coverage further. Doing so produces our highest F-score overall (0.688).

A detailed error analysis of the results produced by $score_{\text{socal+pat+pp}}$ reveals that of the 46 questions it got wrong, 7 (15%) were due to one or both adjectives being OOV, 11 (24%) were questions where the adjective in the answer was modified by an intensifying adverb, which was not handled by our decision procedure, 4 (9%) were cases where the question and answer adjectives were synonymous, and the rest (roughly 50%) were caused by incorrect polarity predictions. Therefore, further gains on this task might be made by modifying the decision procedure to handle adverb modifiers, and improving the accuracy of the adjective intensity prediction method.

4.6. Conclusion

The pivot method used to extract bilingually-induced paraphrases is built on top of techniques for phrase-based machine translation. As a result, some paraphrases in PPDB are composed of a single-word term on one side, and a multi-word phrase that respects sentence constituent boundaries on the other. This, coupled with their wide coverage, makes bilingually-induced paraphrases uniquely useful for studying the meaning of compositional phrases at scale.

In this chapter, we focused on adjectival phrase paraphrase pairs as a source of information for inferring relative scalar adjective intensity. We found that this paraphrase-based intensity evidence produces pairwise predictions that are less precise than those produced by pattern- or lexicon-based evidence, but with substantially higher coverage. Thus paraphrases can be successfully used as a complementary source of information for reasoning about adjective intensity.

This finding supports one of the central themes of this thesis – that paraphrase-based signals can be combined effectively with other types of features to produce robust models of lexical semantics.

CHAPTER 5 : Extracting Sense-specific Examples of Word Use via Bilingual Pivoting

5.1. Introduction

The previous two chapters examined ways in which signals, or features, derived from bilingually-induced paraphrases can be used directly in models for lexical semantic tasks. We saw that while paraphrase-based signals were reasonably effective for discriminating word sense and predicting scalar adjective intensity on their own, in both cases the model accuracy improved when combining paraphrase features with monolingually-extracted features like contextual similarity and lexico-syntactic patterns. In this way, the bilingually- and monolingually-induced features were shown to be complementary.

This chapter shifts focus toward a method for using paraphrases to build sense-tagged corpora which can then be used to train models for sense-aware tasks. Namely, we exploit bilingual pivoting (Bannard and Callison-Burch, 2005) – the same technique used to extract PPDB – as a means to extract sense-specific examples of word and phrase usage. This chapter details the process of extracting sentences for a target word pertaining to a particular sense. In the next chapter, we will use these sense-specific contexts to train models for tasks where contextualized meaning is important.

5.2. Motivation

Firth famously said that "*The complete meaning of a word is always contextual, and no study of meaning apart from a complete context can be taken seriously*" (Firth, 1935). While lexical semantic tasks, such as relation prediction, have been studied extensively in a non-contextual setting (as we did in Chapters 3 and 4), applying such models to a downstream task like textual inference or question answering requires taking the full context into account. For example, it may be true that a *flower* is a type of *plant*, but *flower* is not within the realm of possible answers to the question *"Which $\underline{\underline{\text{plant}}}$ will GM close next year?"*

Many NLP tasks are built around the challenge of inferring meaning within a specific context. Word sense induction, for example, aims to enumerate the potential different meanings of a given phrase within a large corpus (Navigli, 2009), and typically does so by comparing the contexts of each phrase instance. A recent trend in representation learning is to model semantics at the word sense level rather than the word type level, either via continuous models of contextual meaning (Peters et al., 2017, 2018; Devlin et al., 2019), or discrete sense representations (Reisinger and Mooney, 2010; Huang et al., 2012; Neelakantan et al., 2014; Chen et al., 2014; Guo et al., 2014; Li and Jurafsky, 2015; Kawakami and Dyer, 2015; Mancini et al., 2017; Šuster et al., 2016; Upadhyay et al., 2017). Even within the field of semantic relation prediction, some work has moved beyond the traditional non-contextual task to a study of predicting semantic relations in context (Huang et al., 2012; Shwartz and Dagan, 2016a; Vyas and Carpuat, 2017).

It can be a challenge to develop corpora for training models for tasks where contextualized word meaning is important, since particular attention must be paid to making sure the distribution of instances for a given word reflects its various meanings. Previous approaches to constructing sense-aware corpora include manual annotation (Edmonds and Cotton, 2001; Mihalcea et al., 2004; Hovy et al., 2006; Weischedel et al., 2013), the use of existing lexical semantic resources like WordNet (Miller, 1995; Vyas and Carpuat, 2017), supervised sense tagging using word sense disambiguation systems (Ando, 2006; Zhong and Ng, 2010; Rothe and Schütze, 2015), or unsupervised sense tagging based on foreign word alignments (Gale et al., 1992; Dagan and Itai, 1994; Diab and Resnik, 2002; Ng et al., 2003; Lefever et al., 2011).

This chapter proposes a new method for compiling sense-specific instances of word use in a fully automatic way, inspired by the bilingual pivoting technique used to extract paraphrases in PPDB (Bannard and Callison-Burch, 2005; Ganitkevitch et al., 2013; Pavlick et al., 2015b). Our approach is based on the idea that the many fine-grained senses of a word are instantiated by its paraphrases. For example, the word *plant* has different meanings corresponding to its paraphrases *vegetable*, *installation*, and *factory.* Our method enables the automatic extraction of sentences containing *plant* in its *factory* sense ("*The inspection commission visited a graphite plant and a missile engine testing facility...*") or its *vegetable* sense ("*We have seen the first genetically modified plant raw goods arrive in Europe*"). Unlike other unsupervised methods for sense tagging that use foreign translations as proxies for sense labels, our method uses same-language paraphrases to denote sense, and exploits their shared translations to extract sense-specific sentences from bitext corpora. Because we use same-language paraphrases as sense labels, it is straightforward to map the extracted sentences to existing sense inventories.

The automatic sense-tagging method we describe in Section 5.4.3 is applied to produce a new resource called Paraphrase-Sense-Tagged Sentences (PSTS), which contains up to 10,000 sentences for each of the 3 million highest-quality lexical and phrasal paraphrase pairs in PPDB 2.0 (Pavlick et al., 2015b). In Section 5.5.3, the sentences in PSTS are evaluated by humans based on how 'characteristic' they are of the paraphrase meaning, and we describe a method for re-ranking the sentences to correlate with human judgments of sentence quality. Chapter 6 builds on this chapter by demonstrating potential uses of PSTS for training models for sense-aware tasks.

5.3. Methods for Sense Tagging

In general, there are three basic categories of techniques for generating sense-tagged corpora: manual annotation, the application of supervised models for word sense disambiguation, and unsupervised methods. Manual annotation asks humans to hand-label word instances with a sense tag, assuming that the word's senses are enumerated in an underlying sense inventory (typically WordNet) (Petrolito and Bond, 2014). Manually sense-tagged corpora, such as SemCor (Miller et al., 1994) or OntoNotes (Weischedel et al., 2013), can then be used to train supervised word sense disambiguation (WSD) classifiers to predict sense labels on untagged text. Top-performing supervised WSD systems achieve roughly 74% accuracy in assigning WordNet sense labels to word instances (Ando, 2006; Rothe and Schütze, 2015). In shared task settings, supervised classifiers generally out-perform unsupervised WSD systems (Mihalcea et al., 2004).

Within the set of unsupervised methods, one of the most prolific ideas is to use foreign translations as proxies for sense labels of polysemous words (Brown et al., 1991; Dagan, 1991) (see Section 2.2.2). This is based on the assumption that a polysemous English word e will have different translations into a target language, depending on the sense of e that is used. To borrow an example from Gale et al. (1992), if the English word $e =$*sentence* is translated to the French $f =$*peine* (judicial sentence) in one context and the French $f' =$*phrase* (syntactic sentence) in another, then the two instances in English can be tagged with appropriate sense labels based on a mapping from the French translations to the English sense inventory. This technique has been frequently applied to automatically generate sense-tagged corpora, in order to overcome the costliness of manual sense annotation (Gale et al., 1992; Dagan and Itai, 1994; Diab and Resnik, 2002; Ng et al., 2003; Chan and Ng, 2005; Lefever et al., 2011). Our approach to unsupervised sense tagging in this chapter is related, but different. Like the translation proxy approach, our method relies on having bilingual parallel corpora. But in our case, the sense labels are grounded in English paraphrases, rather than in foreign translations. This means that our method does not require any manual mapping from foreign translations to an English sense inventory. It also enables us to generate sense-tagged examples using bitext over multiple pivot languages, without having to resolve sense mapping between languages.

5.4. Generating Paraphrase-Sense-Tagged Sentences

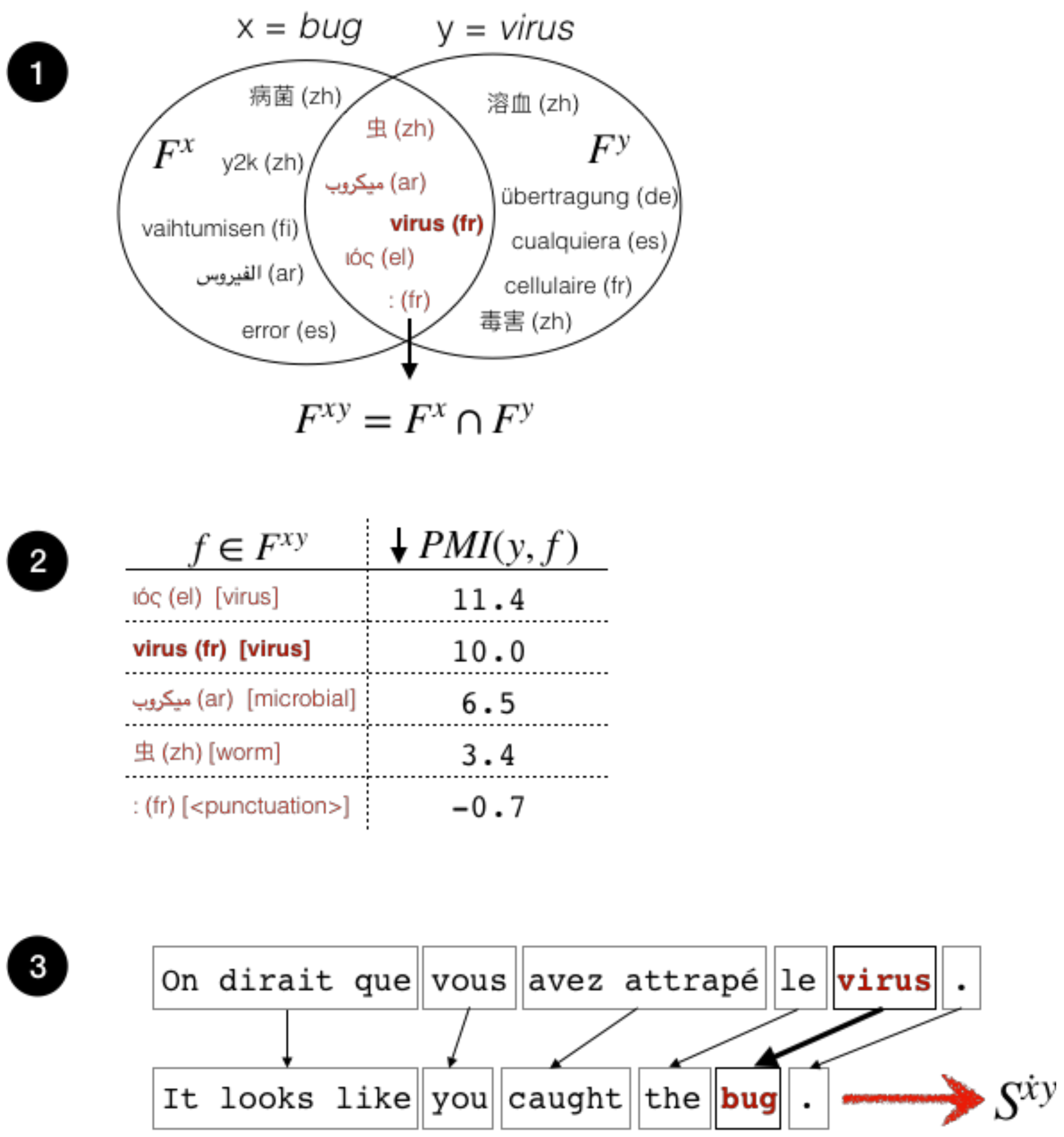
Here we propose a method for exploiting bilingual pivoting (Bannard and Callison-Burch, 2005) to construct a large dataset of sense-specific phrase instances in context. Bilingual pivoting discovers same-language paraphrases by ‘pivoting’ over bilingual parallel corpora. Specifically, if two English phrases such as *coach* and *trainer* are each translated to the same Slovenian phrase *trener* in some contexts, then this is taken as evidence that *coach* and *trainer* have approximately similar meaning. We use this idea in reverse: if two English phrases are known to have similar meaning (i.e. are paraphrases), we find the translations they share in common, and find sentences in bitext corpora where each phrase has been aligned to one of their common translations. For example, given the paraphrase pair *coach* $\leftrightarrow$ *trainer*, if we want to find an English sentence where *coach* means *trainer* (as opposed to *bus* or *railcar*), we look for sentences in English-Slovenian parallel corpora where *coach* has been aligned to their common translation *trener*.

The general process for extracting PSTS sentences for PPDB paraphrase pair $x \leftrightarrow y$ from the English side of English-to-foreign bitext corpora is as follows.[1] Because the pair $x \leftrightarrow y$ is in PPDB, and PPDB was extracted using the pivot method, we can assume there exists some set F^{xy} of foreign phrases to which x and y have both been independently translated. To find sentences containing x that correspond to its sense as a paraphrase of y, we simply enumerate English sentences containing x from the parallel corpora where x is aligned to some $f \in F^{xy}$. Sentences for y are extracted in the same way. We refer to the set of English sentences containing x in its sense as a paraphrase of y as $S^{\dot{x}y}$, and the set of English sentences containing y in its x sense as $S^{x\dot{y}}$. Note that for some other paraphrase pair involving x, say $x \leftrightarrow z$, there may be sentences that appear in both $S^{\dot{x}y}$ and $S^{\dot{x}z}$ if their sets of shared translations, F^{xy} and F^{xz}, overlap. The process is illustrated in Figure 24, and described in further detail below.

5.4.1. Step 1; Finding Shared Translations

In order to find sentences containing the English term x where it takes on its meaning as a paraphrase of y, we begin by finding the sets of foreign translations for x and y, F^x and F^y respectively. These translations are enumerated by processing the phrase-based alignments induced between English sentences and their translations within a large, amalgamated set of English-to-foreign bitext corpora. Once the translation sets F^x and F^y are extracted for the individual terms, we take their intersection as the set of shared translations, F^{xy}.

[1]Note that $x \leftrightarrow y$ is characterized by both the lexicalizations (word or phrase) of x and y, and a shared part-of-speech tag (for words) or sentence constituent label (for phrases). For example, the paraphrase pair *NN: bug* $\leftrightarrow$ *bother* is separate from the pair *VB: bug* $\leftrightarrow$ *bother*.



$f \in F^{xy}$	$\downarrow PMI(y, f)$
ιός (el) [virus]	11.4
virus (fr) [virus]	10.0
ميكروب (ar) [microbial]	6.5
虫 (zh) [worm]	3.4
: (fr) [<punctuation>]	-0.7

On dirait que | vous | avez attrapé | le | **virus** | .

It looks like | you | caught | the | **bug** | .

Figure 24: Diagram of the process for extracting sentences containing the noun x =*bug* in its y =*virus* sense from parallel corpora for PSTS set $S^{\dot{x}y}$. In Step (1), the set of translations shared by *bug* and *virus* is enumerated and named F^{xy}. In Step (2), the translations $f \in F^{xy}$ are ranked by $PMI(y, f)$, in order to prioritize *bug*'s translations most 'characteristic' of its meaning in the *virus* sense. In Step (3), sentences where *bug* has been aligned to the French translation f =*virus* are extracted from bitext corpora and added to the set $S^{\dot{x}y}$.

5.4.2. Step 2: Prioritizing Translations to Produce Characteristic Sentences

Our goal is to build $S^{\dot{x}y}$ such that its sentences containing x are "highly characteristic" of x's shared meaning with y, and vice versa. However, not all pivot translations $f \in F^{xy}$ produce equally characteristic sentences. For example, consider the paraphrase pair *bug* $\leftrightarrow$ *worm*. Their shared translation set, $F^{bug,worm}$, includes the French terms *ver* (*worm*) and *espèce* (*species*), and the Chinese term 虫 (*bug*). In selecting sentences for $S^{b\dot{u}g,worm}$, PSTS should prioritize English sentences where *bug* has been translated to the most characteristic translation for *worm* – *ver* – over the more general 虫 or *espèce*.

The degree to which a foreign translation is "characteristic" of an English term can be quantified by the pointwise mutual information (PMI) of the English term with the foreign term, based on the statistics of their alignment in bitext corpora. To avoid unwanted biases that might arise from the uneven distribution of languages present in our bitext corpora, we treat PMI as language-specific. Given language l containing foreign words $f \in l$, we use shorthand notation f_l to indicate that f comes from language l. The PMI of English term e with foreign word f_l can be computed as:

$$\text{PMI}(e, f_l) = \frac{p(e, f_l)}{p(e) \cdot p(f_l)} = \frac{p(f_l|e)}{p(f_l)} \tag{5.1}$$

The term in the numerator of the rightmost expression is the translation probability $p(f_l|e)$, which indicates the likelihood that English word e is aligned to foreign term f_l in an English-l parallel corpus. Maximizing this term promotes the most frequent foreign translations for e. It is calculated as:

$$p(f_l|e) = \frac{\text{count}(e \to f_l)}{\sum_{f' \in l} \text{count}(e \to f')} \tag{5.2}$$

where $(e \to f_l)$ indicates the event that e is aligned to f_l in a bitext sentence pair.

The term in the denominator is the likelihood of the foreign word, $p(f_l)$. Dividing by this term down-weights the emphasis on frequent foreign words. This is especially helpful for mitigating errors due to mis-alignments of English words with foreign stop words or punctuation. The foreign word probability is calculated as:

$$p(f_l) = \frac{\text{count}(f_l)}{\sum_{f' \in l} \text{count}(f')} \tag{5.3}$$

5.4.3. Step 3: Extracting Sentences

To extract $S^{\dot{x}y}$, the set of English sentences containing x for paraphrase pair $x \leftrightarrow y$, we first order their shared translations, $f \in F^{xy}$, by decreasing $PMI(y, f)$. Then, for each translation f in order, we extract up to 2500 sentences from the bitext corpora where x is translated to f. This process continues until $S^{\dot{x}y}$ reaches a maximum size of 10k sentences.[2] As a result of selecting sentences containing x in decreasing order of $PMI(y, f)$, we intend for PSTS to include contexts where the sense of x is most closely related to its paraphrase y. Table 14 gives examples of sentences extracted for various paraphrases of the adjective *hot*, ordered by decreasing PMI.

$(x \leftrightarrow y)$	f	$\log p(f\|y)$	$\log p(f)$	PMI(y, f)	Sentence segment
	cálida (es)	-1.96	-12.75	10.79	With the end of the hot season last year, ...
hot $\leftrightarrow$ warm	ciepłego (pl)	-3.92	-14.34	10.42	I think that a hot cup of milk...would be welcome.
	chaudes (fr)	-3.30	-12.63	9.33	Avoid getting your feet too close to hot surfaces...
	吃辛辣 (zh)	-4.41	-17.75	13.34	People...should shun hot dishes.
hot $\leftrightarrow$ spicy	épicé (fr)	-1.61	-14.32	12.72	Hot jambalaya!
	pimentés (fr)	-5.75	-17.34	11.59	Get your red hot pu-pus!
	en vogue (fr)	-7.32	-16.46	9.14	Ross is so hot right now.
hot $\leftrightarrow$ popular	très demande (fr)	-9.11	-17.47	8.36	This area of technology is hot.
	热门 (zh)	-3.61	-11.77	8.17	Now the town is a hot spot for weekend outings.

Table 14: Example PSTS sentence segments for the adjective x=*hot* as a paraphrase of $y \in \{\text{warm}, \text{spicy}, \text{popular}\}$. For each example, the pivot translation f is given along with its translation probability $p(f|y)$, foreign word probability $p(f)$, and PMI(y, f).

PSTS is extracted from the same English-to-foreign bitext corpora used to generate En-

[2]Note that this process means that for some frequent English words, PSTS contains sentences pertaining to only four different translations.

POS	Paraphrase pairs	Mean $\lvert S^{\dot{x}y} \rvert$	Median $\lvert S^{\dot{x}y} \rvert$
N*	1.8M	856	75
V*	1.1M	972	54
R*	0.1M	1385	115
J*	0.3M	972	72
Total	3.3M	918	68

Table 15: Number of paraphrase pairs and sentences in PSTS by macro-level part of speech (POS). The number of sentences per pair is capped at 10k in each direction.

glish PPDB (Ganitkevitch et al., 2013), consisting of over 106 million sentence pairs, and spanning 22 pivot languages. Sentences are extracted for all paraphrases as needed to cover the vocabulary in the experiments in Sections 6.2.3-6.4.4, as well as all paraphrases with a minimum PPDBSCORE threshold of at least 2.0. The threshold value serves to produce a resource corresponding to the highest-quality paraphrases in PPDB, and eliminates considerable noise. In total, sentences were extracted for over 3 million paraphrase pairs covering nouns, verbs, adverbs, and adjectives (21 part-of-speech tags total). Table 15 gives the total number of paraphrase pairs covered and average number of sentences (combined for both phrases) per pair. Results are given by macro-level part-of-speech, where, for example, N* covers part-of-speech tags NN, NNS, NNP, and NNPS, and constituent tag NP.

5.5. Evaluating and Re-Ranking PSTS

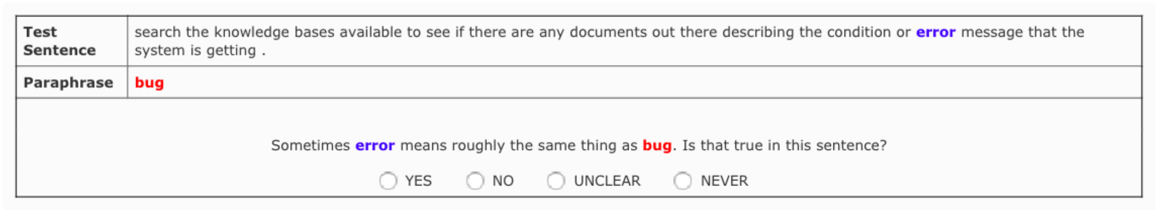
In order to assess the quality of the resource we solicit human judgments. There are two primary questions to address:

- Do automatically-extracted PSTS sentences for a paraphrase pair truly reflect the shared sense of that paraphrase pair? Specifically, for sentences like s_{bug} where $s_{bug} \in S^{\dot{bug},virus}$, does the meaning of the word *bug* in s_{bug} actually reflect its shared meaning with *virus*?

- How well does the PMI-based unsupervised ranking method correlate with human judgments of contextual similarity? If we draw a random sentence s_{bug} from $S^{\dot{bug},virus}$, which was generated by pivoting over foreign translation f, does the value $PMI(\text{virus}, f)$ actually tell us how similar the meaning of *bug* is to *virus* in this sentence?

5.5.1. Human annotation setup

To investigate these questions, we ask humans to evaluate how characteristic PSTS sentences are of their corresponding paraphrase pair. Specifically, for a paraphrase pair like *bug*↔*insect*, annotators are presented with a sentence containing *bug* from $S^{\dot{bug},insect}$, and asked whether *bug* means roughly the same thing as *insect* in the sentence. We repeat the process in the other direction, showing annotators sentences containing *insect* from $S^{bug,\dot{insect}}$, and asking them whether *insect* means roughly the same thing as *bug* in each case. The annotators can choose from responses *yes* (the meanings are roughly similar), *no* (the meanings are different), *unclear* (there is not enough contextual information to tell), or *never* (these words can never have similar meaning). We instruct annotators to ignore grammaticality in their responses, and concentrate specifically on the semantics of the paraphrase pair. An example annotation instance within the user interface is shown in Figure 25.



Test Sentence	search the knowledge bases available to see if there are any documents out there describing the condition or **error** message that the system is getting .
Paraphrase	**bug**

Sometimes **error** means roughly the same thing as **bug**. Is that true in this sentence?

Figure 25: Screenshot of a single annotation instance for the sentence-paraphrase pair (s_{error}, bug).

Human annotation is run in two rounds, with the first round of annotation completed by NLP researchers, and the second (much larger) round completed by crowd workers via Amazon Mechanical Turk (MTurk). Responses from the first round of annotations are used to construct 'control' instances to gauge worker accuracy and agreement in the second round.

In the first round of annotation (done by NLP researchers), sentence-paraphrase instances are generated for annotation as follows. We begin with a list of 40 hand-selected polysemous target words (10 each of nouns, verbs, adjectives, and adverbs). For each target word x, there are 3 paraphrases y randomly selected from PPDB (two lexical and one phrasal).[3] Next, for each paraphrase pair $x \leftrightarrow y$, we randomly select three sentences from PSTS containing the target word x, $s_x \in S^{\dot{x},y}$, and use them to form sentence-paraphrase annotation instances (s_x, y). Instances are also generated for each paraphrase pair in the reverse direction, selecting three sentences containing y, $s_y \in S^{x,\dot{y}}$, to form annotation instances (s_y, x). Of the 720 total instances generated in this way, we randomly select a batch of 240 to present to researchers for annotation. The actual annotation is carried out by a group of 10 annotators, split into 5 teams of 2. To encourage consistency, each pair of annotators works together to annotate each instance. For redundancy, we also ensure that each instance is annotated separately by two pairs of researchers. In this first round, the annotators have inter-pair agreement of 0.41 Fleiss' kappa (after mapping all *never* answers to *no*), indicating weak agreement (Fleiss, 1971).

In the second round we follow a similar method for generating instances for annotation. Starting with the same set of 40 target words, there are now 4 paraphrases (3 lexical, 1 phrasal) selected randomly from PPDB for each target. For each $x \leftrightarrow y$ paraphrase pair, we randomly select 4 sentences from PSTS in each direction. Of the 1280 sentence-paraphrase instances generated, we randomly choose 1000 total for annotation. Each instance is evaluated individually by 7 workers on MTurk. In each MTurk assignment, we also include one of the instances from round one that was annotated as unanimously *yes* or unanimously *no* by the NLP researchers in order to gauge agreement between rounds. In round two, the annotators have inter-annotator agreement of 0.33 Fleiss' kappa (after mapping all *never* answers to *no*), which is slightly lower than that of the NLP researchers in round 1. The crowd workers had 75% absolute agreement with the 'control' instances inserted from the previous round.

[3]In order to promote high-quality paraphrase pairs, we randomly select from paraphrases in PPDB having a PPDB2.0 Score of at least 2.0 (for lexical paraphrases) or 3.0 (for phrasal paraphrases)

5.5.2. Human annotation results

In order to assess the quality of sentences in the PSTS resource, we measure the average annotator score for each instance, where *no* and *never* answers are mapped to the value 0, *yes* answers are mapped to the value 1, and *unclear* answers are ignored (because the annotator indicated there was not enough contextual information to make a decision). The combined results of this calculation from both rounds are given in Table 16.

Overall, the average score of each instance is 0.63, indicating that more sentence-paraphrase instances from PSTS are judged by humans to have similar meaning than dissimilar meaning. The results vary by part of speech, and whether the paraphrases involved are lexical (i.e. single word) or phrasal. In general, adjectives produce higher-quality PSTS sentences than the other parts of speech. For nouns and adjectives, phrasal paraphrase pairs are judged to have higher quality than lexical paraphrase pairs. For verbs and adverbs, the results are reversed.

POS	Lexical/Phrasal	Avg.Rating
NN	Lexical	0.57
	Phrasal	0.67
VB	Lexical	0.66
	Phrasal	0.51
JJ	Lexical	0.69
	Phrasal	0.73
RB	Lexical	0.67
	Phrasal	0.37
Total	Combined	0.63

Table 16: Human evaluation of contextual similarity of sentence pairs.

Given that there is such variation in the quality of PSTS sentences, it would be useful to have a metric that indicates quality. In the formation of PSTS, we used the point-wise mutual information $PMI(y, f)$ of the English paraphrase y with the shared foreign translation f as an indicator for how characteristic a sentence containing English target word x is of its shared meaning with y. Here we evaluate whether that was actually a good metric, by measuring the Spearman correlation (Appendix A.1) between the PMI metric and averaged annotated human judgements of sentence-paraphrase quality. The results for this calculation are given in Table 17.

POS	ρ	p-value
NN	0.12	0.04
VB	0.26	< 0.01
JJ	0.17	< 0.01
RB	0.29	< 0.01
Combined	0.22	< 0.01

Table 17: Spearman correlation (ρ) between PMI and average human rating of contextual similarity for each sentence.

The Spearman correlation between the PMI metric and the average human rating for each sentence-paraphrase instance was 0.22 ($p < 0.01$), indicating only a weak positive correlation. In order to analyze why this is the case, we qualitatively examined instances that have high PMI but low human rating (first case) and vice versa (second case). Table 18 shows examples for each of these cases.

Case	Reason	Example					
		Target	Paraphrase	Sentence	Translation	PMI	Rating
High PMI, Low Rating	More specific paraphrase	tight	watertight	...ensure the room is light tight.	étanche (fr)	11.8	0.0
	Opposite ADVP	really	not at all	Kate was really upset when you made your choice to come with us.	pas du tout (fr)	12.0	0.1
Low PMI, High Rating	Polysemous paraphrase	bureau	board	...a senior official with the Beijing health bureau said Friday.	مصلحة (ar)	0.7	1.0

Table 18: Examples of annotated instances where the PMI between the paraphrase and shared translation did not correlate with the human rating.

In the first case, we examined ten target sentence-paraphrase instances that had an average human rating below 0.2, and PMI of the English paraphrase with the shared foreign translation more than 11.4, or 1.5 standard deviations above the mean (the PMI values were approximately normally distributed over instances, with mean 6.3 and standard deviation 3.4). Examining these instances with high PMI but low human rating indicated two trends. In eight of the ten cases, the paraphrase could be classified as a rarer and more specific instance of the target word, and the shared foreign translation was also relatively rare (i.e. had a smaller than average translation probability). The more specific paraphrase was not an appropriate substitute for the target sentence in these cases, leading to a low human rating. But because probability $p(f)$ of the shared translation was low and the alignment probability between the translation and the paraphrase, $p(f|e)$, was relatively high (as specific words tend to have fewer possible translations than general words), the PMI score $PMI(e, f) = \frac{p(f|e)}{p(f)}$ was high. The other two instances with high PMI but low human rating were both ADVP paraphrase pairs, which were semantically opposites but likely extracted as paraphrases via biligual pivoting due to instances where one of the adverbial modifiers was translated in an opposite way. For example, the adverb phrases (*really* $\leftrightarrow$ *not at all*) are PPDB paraphrases, and may share an aligned foreign phrase like the French *pas du tout* if, for example, *really upset* has been translated as *pas du tout joyeux* or *not at all happy.*

In the second case, we examined fifteen target sentence-paraphrase instances that had an average human rating above 0.8, and PMI value less than 1.2, or 1.5 standard deviations below the mean. The vast majority of these had low PMI driven by a low alignment probability $p(f|e)$, due to the paraphrase e being a polysemous word whose most frequent sense is something other than the target.

5.5.3. Supervised Sentence Quality Ranking

Although PMI was used as a sentence quality indicator when extracting sense-specific sentences for each paraphrase pair, our analysis indicates that PMI is only weakly correlated with human judgements of sentence quality. In order to enable selection within PSTS of the most characteristic sentences for each paraphrase pair for downstream tasks, this section describes a model to re-rank PSTS sentences in a way that better correlates with human judgements of their quality.

Our goal is to train a model that can predict the average human quality rating for a target sentence-paraphrase instance. Concretely, given a target word x, its paraphrase y, and a sentence $s_x \in S^{\dot{x},y}$ extracted by pivoting over translation f, the model should predict a score whose magnitude indicates how characteristic s_x is of x's shared meaning with y.

We formulate this as ordinary least squares linear regression, where the dependent variable is the quality rating and the features are computed based on the input. There are four groups, or types, of features used in the model:

- **PPDB Features.** For paraphrase pair $x \leftrightarrow y$, there are seven corresponding features from PPDB 2.0 used as input to the model. These correspond to the pair's PPDBSCORE, and six additional features concerning translation and paraphrase probabilities.

- **Contextual Features.** The three contextual features are designed to measure the distributional similarity between the target x and paraphrase y, as well as the substitutability of paraphrase y for the target x in the given sentence. They include the mean cosine similarity between paraphrase y and tokens within a two-word context window of x in sentence s_x; the cosine similarity between context-masked embeddings for x and y in s_x (using the method of Vyas and Carpuat (2017) – see Section 2.3.3); and the AddCos lexical substitutability metric where y is the substitute, x is the target, and the context is extracted from s_x (Eq. 3.16) (Melamud et al., 2015b).

- **Syntactic Features.** There are five binary indicator features used to indicate the coarse part-of-speech label assigned to paraphrase pair $x \leftrightarrow y$ (NN, VB, RB, or JJ), and whether $x \leftrightarrow y$ is a lexical or phrasal paraphrase pair.

- **PMI.** The final feature is simply $PMI(y, f)$ (Eq. 5.1).

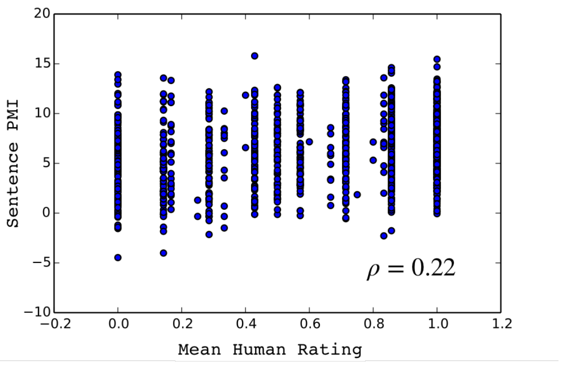
The features used as input to the model training process are the sixteen listed above, as well as their interactions as modeled by degree-2 polynomial combinations (153 features total). During training and validation, we apply feature selection using recursive feature elimination in cross-validation as detailed below.

The dataset available for training and evaluating the model is composed of the 1227 target sentence-paraphrase instances that were annotated in one or both rounds of human evaluation, after ignoring instances marked as 'unclear' by two or more workers. The quality rating for each instance is taken as the average annotator score, where *no* and *never* answers are mapped to the value 0, *yes* answers are mapped to the value 1, and *unclear* responses are ignored.

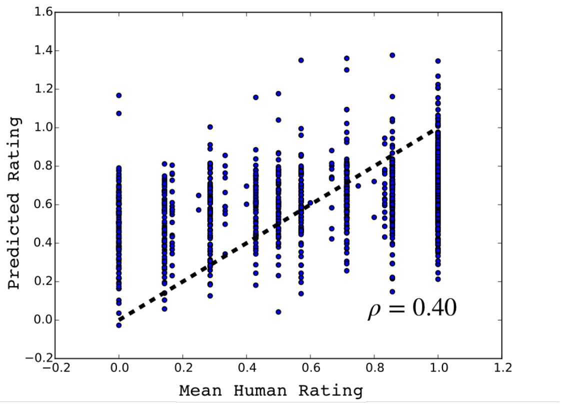
Due to the limited size of the dataset, we first use cross-validation to estimate the model reliability, and subsequently re-train the linear regression model on the entire set of instances for weighting sentences in PSTS. For model evaluation, we run 5-fold cross-validation. In each fold, we first run recursive feature elimination with cross-validation (RFECV) (Guyon et al., 2002) on the training portion, then train a linear model on the selected features and predict ratings for the test portion. The predicted ratings on held-out portions from each fold are compared to the mean annotator ratings, and Spearman correlation is calculated on the combined set of all instances (Figure 26b).

The resulting correlation between predicted and human ratings is 0.40, which is substantially higher than the correlation of 0.22 between target sentence PMI and human ratings. Additionally, while a correlation of 0.40 is not very high, it is important to note that the correlation between each individual annotator and the mean of other annotators over all target sentence-paraphrase instances was only 0.37. Thus the model predicts the mean annotator rating with roughly the same reliability as individual annotators.

Finally, we re-train the regression on the entire dataset of target-sentence instances (again using RFECV to select features). This model can be used to score and re-rank all sentences in the PSTS resource. In the chapter that follows, we refer to the score produced by this model as the sentence *quality score*.



(a) Correlation between PMI and average human ratings



(b) Correlation between model-predicted and average human ratings

Figure 26: The Spearman correlation between sentence PMI and average human rating is $\rho = 0.22$ (a); by using linear regression to predicts average sentence ratings, the correlation increases to $\rho = 0.40$ (b).

5.6. Conclusion

In Chapter 3 we assumed that the various meanings of a word could be modeled by discrete sense clusters, and presented a method for partitioning the paraphrases of a target word into clusters representing its coarse senses. In this chapter, we took the more extreme view that the fine-grained senses of a word are instantiated by its paraphrases. We applied this idea to the challenge of automatically building sense-tagged corpora. The proposed method adapts bilingual pivoting (Bannard and Callison-Burch, 2005) to extract paraphrase-specific examples of target words automatically from the English side of bilingual parallel corpora.

Our proposed method was used to produce a dataset called Paraphrase-Sense-Tagged Sentences (PSTS) containing sentence-level contexts for over 3M paraphrase pairs from PPDB (Ganitkevitch et al., 2013; Pavlick et al., 2015b). The quality of sentences in PSTS was evaluated by humans, who indicated that the majority of sentences pertaining to a paraphrase pair were reflective of the shared meaning of that pair. In order to enable the selection of the highest-quality sentences from PSTS, we also trained a regression model to predict the human quality rating of each sentence.

One of the limitations of the work in this chapter is that we avoided a direct comparison between the sense-specific word usage examples in PSTS, and those that might be produced using a pre-trained word sense disambiguation model. The advantages of using bilingual pivoting, rather than a WSD model, to extract sense-specific contexts for target words are that the process does not require an underlying sense inventory, making it flexible, and is completely unsupervised. We leave the direct comparison between PSTS and a hypothetical resource produced using a WSD model for future work.

CHAPTER 6 : Applications of Sense-specific Examples of Word Use

6.1. Introduction

This chapter extends the previous one by demonstrating the ability to use sense-specific word instances from the Paraphrase-Sense-Tagged Sentences (PSTS) dataset as a training bed for three lexical semantic tasks: fine-grained sense embeddings, word sense induction, and contextual relation prediction.

In the first case, we take the view that a word has as many fine-grained senses as it has paraphrases, and we use PSTS as the basis for generating fine-grained sense (paraphrase-level) embeddings for terms in PPDB. We describe two different methods for training paraphrase embeddings over PSTS. Then, we evaluate the embeddings produced by each method on a set of semantic similarity and relatedness benchmarks, and compare the performance of each paraphrase-embedding method to a counterpart embedding model at the word type level. We show that the paraphrase embeddings do a better job at capturing semantic similarity than their word embedding counterparts.

In the second application, we describe a method for word sense induction that assumes the PPDB sense clusters from Chapter 3 as a sense inventory, and uses the paraphrase embeddings from PSTS to map word instances onto the most appropriate sense. The method is shown to produce competitive results on two existing shared task datasets.

Finally, we use the PSTS sentences corresponding to known hypernym-hyponym pairs to automatically generate training data for a contextual hypernym prediction model. The dataset created is five times larger than existing datasets for this task. We train a contextual hypernym prediction model on this PSTS dataset, and show that it leads to more accurate predictions than the same model trained on a smaller, hand-labeled training set.

6.2. Applications 1: Paraphrase Embeddings

In some applications, encoding terms at the type level may be too general because an individual word or phrase can mean multiple things. Having one vector per phrase type means that we cram all the meanings of a phrase into a single vector, which can be problematic. For instance, when clustering paraphrases of *bug* by the sense of *bug* they convey, the vector for *bug*'s paraphrase *mike* encodes both its male-name sense and audio sense. Clustering algorithms fail to cluster *mike* with *microphone* because the embedding for *mike* is dominated by its name sense.

At a very fine-grained level, we might say that a given word has as many senses as it has paraphrases; the word *bug* has slightly different senses when understood to be a paraphrase of *microphone*, *insect*, or *mosquito*, although clearly the latter two meanings are related. By assigning a different vector to each of *bug*'s paraphrase-level senses, we hope to capture the variety of semantic meaning attributable to each of *bug*'s paraphrases.

This section proposes two approaches for generating paraphrase-level embeddings based on the sentences available in PSTS. The first is based on the *skip-gram* word embedding model (Mikolov et al., 2013b), and the second is based on the BERT contextual embedding model (Devlin et al., 2019). For a paraphrase pair $x \leftrightarrow y$, we produce paraphrase embeddings in each direction, $v_{x \to y}$ and $v_{y \to x}$, which each reflect the meaning of the first term in its shared sense with the second. For example, for a paraphrase pair like (bug $\leftrightarrow$ pest), there is an associated paraphrase embedding in each direction: the embedding $v_{\text{bug} \to \text{pest}}$ encodes the meaning of *bug* in its sense as a *pest*, and the embedding $v_{\text{pest} \to \text{bug}}$ encodes the meaning of *pest* in its sense as a *bug*. The embeddings are not equal in both directions because the paraphrase relationship is not necessarily synonymous. In the case of (bug $\leftrightarrow$ pest), for example, *pest* is more general than *bug*. Ideally, the paraphrase-level embeddings should encode this distinction.

6.2.1. Paraphrase-level skip-gram Transfer Embeddings (PP-SG)

The first general approach taken to train paraphrase-level embeddings for paraphrase pairs in PSTS is based on continued training of *skip-gram* embeddings (Mikolov et al., 2013b,a). To train a paraphrase-level embedding such as $v_{\text{bug}\rightarrow\text{pest}}$, we start with a pre-trained *skip-gram* model, and continue training the word embedding for *bug* using its contexts from the PSTS sentences in $S^{bug,pest}$. While running continued training, we hold the context embedding layer fixed and apply the gradient update only to the word embedding for *bug*. The resulting embedding $v_{\text{bug}\rightarrow\text{pest}}$ (which we refer to as **PP-SG**) thus shares the same embedding space as the original pre-trained model, and can be compared directly with word-type embeddings (abbreviated **WT-SG**) in the original pre-trained embedding space.

The equation for updating the paraphrase vector v_w for each input sentence is:

$$v_w^{t+1} = v_w^t - \alpha \cdot \big(\sum_{c \in C} (\sigma(v_c \cdot v_w^t) - 1) \cdot v_c + \sum_{c \in N} \sigma(v_c \cdot v_w^t) \cdot v_c \big) \tag{6.1}$$

where v_w^t is a vector for term w at time t, C is the set of context words appearing within a fixed-width window of w, N is the set of randomly-selected negative sample words (with size $|C| \cdot n$, where n is a tuned parameter), and α is the learning rate. The function σ is the logistic function, i.e. $\sigma(x) = \frac{1}{1+e^{-x}}$.

To train paraphrase-level embeddings for PSTS, we begin with a *skip-gram* model that has been pre-trained on the Annotated Gigaword corpus (Napoles et al., 2012), and includes embeddings for all single- and multi-word phrases in PPDB2.0.[1] The *skip-gram* model has a variety of parameters to be tuned, including the context window size, learning rate, number of negative samples, and epochs. In addition to these, we introduce the continued training

[1]The base *skip-gram* model is trained with the following parameters: context window of size 3, learning rate *alpha* from 0.025 to 0.0001, minimum word count 100, sampling parameter $1e^{-4}$, 10 negative samples per target word, and 5 training epochs. Embeddings for multi-word phrases were generated by replacing each instance of a multi-word phrase from the PPDB vocabulary in the training corpus with single token by substituting spaces for underscores (e.g. `merchant marine` $\rightarrow$ `merchant_marine`).

parameters of maximum number of sentences from S^{xy} (ordered by decreasing quality score) to use for continued training, and minimum PMI of sentences chosen for continued training. All of these parameters are tuned using grid search, where we evaluate each parameter combination on a small hand-crafted development set of target words and their paraphrases using a word sense clustering task. For each of four target words x in (*bug.n*, *film.n*, *hot.j*, and *bright.j*), we train paraphrase-level embeddings $v_{x \to p}$ for paraphrases of x. We then use K-Means to cluster the resulting paraphrase vectors, and evaluate the quality of the predicted clustering as compared to hand-crafted sense clusters using the metrics paired F-Score and V-Measure. We choose the parameter setting that maximizes the sum of F-Score and V-Measure for this test.[2]

This continued skip-gram training approach is carried out for all paraphrases present in PSTS. To qualitatively examine the resulting paraphrase embeddings, Table 19 shows the nearest word-type neighbors in the original model space prior to continued training for words *bug*, *pest*, and *microbe*. Following, Table 20 shows the nearest word-type and paraphrase neighbors in the model space for the corresponding paraphrase embeddings $v_{bug \to pest}$, $v_{pest \to bug}$, $v_{bug \to microbe}$, and $v_{microbe \to bug}$ after continued training.

Intuitively, the continued training process should nudge paraphrase embeddings away from the word-type embedding from which they began (which will be dominated by that word's most frequent sense), and toward the sense indicated by the paraphrase. This is what appears to be happening. For example, the nearest neighbors of the word type *bug* before continued training contain terms related to *bug*'s sense as a computer virus (*viruses, y2k*), but the nearest word-type neighbors for the paraphrase vector $v_{bug \to pest}$ after continued training include only words related to *bug*'s pest sense (*pest*, *insect*, *infestations*, *bug*, and *armyworm*). Likewise, the nearest neighbors of the word type *pest* include different types of pests (e.g. *weed*, *fungus*), while the nearest neighbors of the paraphrase vector $v_{pest \to bug}$ are concentrated closer to bug-type pests (e.g. *insect*, *armyworm*).

[2]The final parameters chosen are a 5-token context window, 5 negative samples ($n = 5$), 10 epochs, initial learning rate alpha=0.25, maximum 150 sentences from $S^{\dot{x}y}$ and $S^{x\dot{y}}$, and minimum PMI 8.0.

Word vector (WT-SG)	Nearest words by WT-SG vector
v_{bug}	bugs, worm, viruses, y2k, infestation
v_{pest}	pests, insect, weed, fungus, infestation
$v_{microbe}$	bacterium, bacteria, parasite, organism, pathogen

Table 19: Nearest neighbors for words *bug*, *pest*, and *microbe* in the original model space, prior to continued training.

Paraphrase vector (PP-SG)	Nearest words by WT-SG vector	Nearest paraphrases by PP-SG vector
$v_{bug \to pest}$	pest, insect, infestations, bug, armyworm	(pest→bug), (bug→worm), (worm→bug), (bug→debugging), (pest→cockroach)
$v_{pest \to bug}$	pest, insect, infestations, armyworm, pests	(bug→pest), (pest→cockroach), (bug→worm), (worm→bug), (bug→debugging)
$v_{bug \to microbe}$	bug, parasite, bacteria, bacterium, microbe	(microbe→bug), (bug→germ), (bug→bacterium), (microbe→germ), (bug→microorganism)
$v_{microbe \to bug}$	microbe, bacteria, bacterium, parasite, microbes	(bug→microbe), (microbe→germ), (bug→germ), (bacterium→bug), (germ→bug)

Table 20: Nearest neighbors for paraphrase-level skip-gram transfer embeddings, after continued training.

6.2.2. Paraphrase-level BERT Embeddings (PP-BERT)

The Bidirectional Encoder Representations from Transformers (BERT) method of Devlin et al. (2019) also provides a convenient mechanism for deriving a context-specific representation for paraphrases. Given a token t_k in context $(t_1, \ldots, t_k, \ldots, t_n)$, BERT produces a vector for t_k that is corresponds to the final hidden layer for that token within a deep bidirectional Transformer encoder (Vaswani et al., 2017).

We use the pre-trained BERT-base (uncased) model[3] to generate paraphrase-specific embeddings (called **PP-BERT**) based on the contexts in PSTS. For a paraphrase pair $x \leftrightarrow y$, we produce vectors $v_{x \to y}$ and $v_{y \to x}$ that are both specifically linked to pair $x \leftrightarrow y$. The vectors are derived from the PSTS contexts $S^{\dot{x}y}$ and $S^{x\dot{y}}$ respectively as follows.

Assume the set of sentences containing x, $S^{\dot{x}y}$, is ranked based on the sentence quality ranking model developed in Section 5.5.3 and truncated to have length at most m (we set $m = 100$). Each sentence $s_i^{\dot{x}y} \in S^{\dot{x}y}$ contains the target word x, and can therefore be used to generate a sentence-specific BERT representation for x.[4] To combine the term embeddings corresponding to all m sentences $s \in S^{\dot{x}y}$, we simply take a weighted average over the m sentences, where the weight ascribed to each sentence is the the quality score for that sentence. Table 22 gives the nearest paraphrase neighbors for four resulting PP-BERT paraphrase embeddings of *bug*.

As we did with the *skip-gram* embeddings, it is useful to have a comparable word type-level BERT embedding model for comparison to form type-level BERT embeddings (abbreviated **WT-BERT**), we randomly select 100 instances of a term x from PSTS sets $S^{\dot{x}\star}$ and take their average. Table 21 gives the nearest word-type neighbors for three terms in WT-BERT embedding space.

[3]`https://github.com/google-research/bert`

[4]If x is a phrase with multiple words, we average the BERT representations for each token in x to get the contextual representation for the phrase x.

Term vector (WT-BERT)	Nearest terms by WT-BERT vector
v_{bug}	bug, a bug, bugging, fault, problem
v_{pest}	pests, pesticide, pest-control, pesticides, the pest
$v_{microbe}$	microbes, microbial, micro-organism, anti-microbial, a bacterium

Table 21: Nearest neighbors for words *bug*, *pest*, and *microbe* in the WT-BERT embedding space.

Paraphrase vector (PP-BERT)	Nearest terms by WT-BERT vector	Nearest paraphrases by PP-BERT vector
$v_{bug \to pest}$	bug, the bug, bugs, bugging, a bug	(bug→animal), (bug→virus), (bug→worm), (bug→debugging), (bug→problem)
$v_{pest \to bug}$	pest, pests, the pest, insect, pest-control	(pest→lice), (pest→cockroach), (pest→larvae), (pest→infection), (pest→parasite)
$v_{bug \to microbe}$	bug, the bug, bugs, the bugs, a virus	(bug→germ), (bug→bacterium), (bug→virus), (bug→thing), (bug→microorganism)
$v_{microbe \to bug}$	microbe, microbes, microbial, micro-organism, micro-organisms	(microbe→germ), (microbe→bacterium), (microbe→organism), (microbe→microorganism), (microbe→micro-organism)

Table 22: Nearest neighbors for paraphrase-level PP-BERT token embeddings.

6.2.3. Intrinsic Evaluation: Predicting Semantic Similarity

Having constructed phrase representations that encode meaning at the paraphrase level, we next test the hypothesis that these paraphrase-level embeddings, which capture a fine-grained sense of a word, capture semantic meaning more precisely than their embedding counterparts at the word type level. For this we evaluate both types of representations via semantic similarity and relatedness prediction, which is frequently used as an intrinsic evaluation method for word embedding quality (Baroni et al., 2014).

The task of semantic similarity prediction is as follows: given two terms x and y (out of context), a system must assign a score that indicates the level of semantic similarity or relatedness that holds between the terms. High scores correspond to high similarity/relatedness and vice versa. The task is evaluated by calculating the correlation of the system's predictions with human-annotated values. Generally, systems compute a predicted value for a word pair (x, y) based on the cosine similarity of their term embeddings, $cos(v_x, v_y)$.

For both types of embeddings generated from PSTS (i.e. *skip-gram* transfer embeddings and BERT paraphrase embeddings), we compare the performance of these paraphrase-level embeddings to their word-type counterparts. This enables us to evaluate the hypothesis that representing terms at the fine-grained paraphrase level leads to more accurate semantic representation. Specifically, we run experiments comparing four different term representation methods:

- *skip-gram* Embeddings
  - **WT-SG**. Word-type embeddings from the pre-trained *skip-gram* model used as the starting point for training PP-SG embeddings.
  - **PP-SG**. The paraphrase-level embeddings produced by continued training of WT-SG embeddings on the top-100 sentences (in terms of PMI) for each paraphrase pair in PSTS.

- BERT Embeddings
  - **WT-BERT**. Word-type embeddings generated by averaging the BERT representation for each term in 100 randomly selected contexts from PSTS.
  - **PP-BERT**. The paraphrase-level embeddings produced by averaging the BERT embeddings for the top-100 sentences (in terms of PMI) for each paraphrase pair in PSTS.

When computing similarity between a pair of terms using paraphrase embeddings, the question of how to select which paraphrases to use to represent each term in the pair naturally arises. Concretely, given term x with paraphrase set $\text{PPS}\textsc{et}(x)$, and term y with paraphrase set $\text{PPS}\textsc{et}(y)$, how do we choose paraphrases $p \in \text{PPS}\textsc{et}(x)$ and $q \in \text{PPS}\textsc{et}(y)$ to represent terms x and y with embeddings $v_{x\to p}$ and $v_{y\to q}$? In these experiments, we compare three methods:

- **Mean Similarity (mean)**. When calculating similarity between terms x and y, we can take the mean cosine similarity between all paraphrase embeddings for x and all paraphrase embeddings for y:

$$avg_{p\in P(x),q\in P(y)}cos(v_{x\to p}, v_{y\to q}) \tag{6.2}$$

- **Maximum Similarity (max)**. When representing terms x and y, we choose the pair of paraphrase embeddings v_x^i, v_y^j that maximize the pairwise cosine similarity between the two terms:

$$max_{p\in P(x),q\in P(y)}cos(v_{x\to p}, v_{y\to q}) \tag{6.3}$$

- **Shortest Path (sp)**. Alternatively, we can use the PPDB graph itself to help disambiguate the terms x and y, and in doing so, select which paraphrase representations

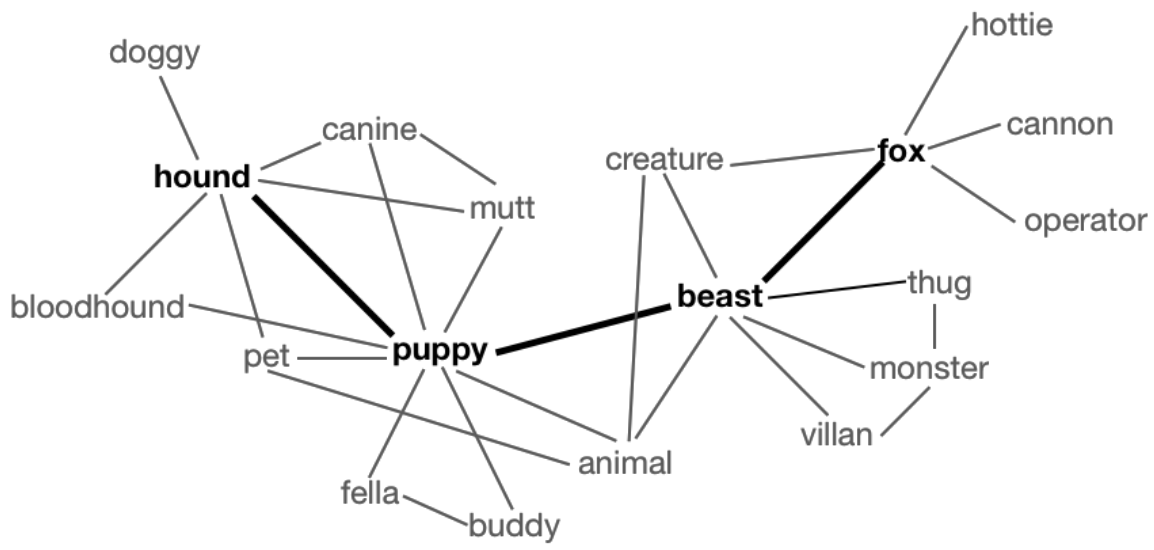


Figure 27: A partial view of the PPDB graph between *fox* and *hound*, with the shortest path highlighted. When calculating similarity between these terms using shortest path paraphrase embeddings, *hound* would be represented using $v_{hound,puppy}$ and *fox* would be represented using $v_{fox,beast}$.

to use. If terms x and y are direct paraphrases of one another in PPDB, we simply use the corresponding embeddings $v_{x \to y}$ and $v_{y \to x}$. If x and y are *not* direct paraphrases in PPDB, but there exists a shortest path between them in the PPDB graph $(x, p, \ldots, q, y)$, then we use the embeddings for paraphrases $x \leftrightarrow p$ and $y \leftrightarrow q$ such that p and q lie directly adjacent to x and y respectively along the shortest path (see Figure 27). To compute the shortest path, we create a graph representation of PPDB, where terms are nodes and edges represent direct paraphrases. We weight each edge (x, y) by the inverse PPDBSCORE for pair $x \leftrightarrow y$.

Semantic similarity and relatedness prediction is run over seven existing benchmark datasets – four containing primarily noun pairs (WS353-SIM, WS353-REL (Finkelstein et al., 2002), MC-30 (Miller and Charles, 1991), and RG-65 (Rubenstein and Goodenough, 1965)), two containing primarily verbs (SimVerb-3500 (Gerz et al., 2016) and YP-130 (Yang and Powers, 2005)), and one containing a mix of parts of speech (SimLex-999 (Hill et al., 2015)). Of the seven benchmarks, all are focused on semantic similarity with the exception of WS353-REL,

		WS353-SIM	WS353-REL	MC-30	RG-65	SIMLEX-999	SimVerb-3500	YP-130
Predominant POS		NN				mix	VB	
Original Dataset Size		203	252	30	65	999	3500	130
Pct Pairs Covered		0.99	0.99	1.00	0.98	0.99	0.92	0.91
Embedding	Method	WS353-SIM	WS353-REL	MC-30	RG-65	SIMLEX-999	SimVerb-3500	YP-130
WT-SG		0.734	0.567	0.671	0.666	0.414	0.398	0.644
PP-SG	mean	0.578	0.426	0.724	0.741	0.482	0.353	0.444
PP-SG	max	0.638	0.438	0.822	0.814	0.601	0.510	0.657
PP-SG	sp	0.648	0.465	0.821	0.748	**0.604**	**0.530**	0.614
WT-BERT		0.700	0.541	0.754	0.786	0.534	0.423	0.653
PP-BERT	mean	0.720	0.559	0.767	0.774	0.533	0.436	0.658
PP-BERT	max	**0.739**	**0.610**	**0.881**	**0.912**	0.589	0.465	**0.707**
PP-BERT	sp	0.721	0.579	0.802	0.786	0.588	0.490	0.673

Table 23: Semantic similarity and relatedness results. For each of 7 datasets, we use the specified Embedding type and paraphrase selection Method to represent pairs for computing semantic similarity or relatedness. Results are given in terms of Spearman correlation with human-annotated ratings.

which assigns scores based on term relatedness (e.g. related terms *hotel* and *reservation* have a high relatedness score, while synonymous terms *midday* and *noon* have a high similarity score).

Table 23 gives the results on the seven benchmarks for each combination of term representation and (for paraphrases) similarity calculation method. Scores are reported in terms of Spearman correlation (ρ) between model predictions and human-annotated similarity scores (Appendix A.1). All correlation coefficients noted are significant ($p \leq 0.001$). The table also lists the percent of pairs covered by all embedding types in each dataset, as some datasets contained words that were out of vocabulary for the *skip-gram* embeddings. For the purpose of direct comparison, any word pair that was out of vocabulary for one or more embedding types was ignored in scoring.

For both the *skip-gram* transfer embeddings and averaged BERT paraphrase embeddings, we find that predicting semantic similarity at the paraphrase level leads to generally better results than doing so at the word-type level, indicating that the paraphrase-level embeddings provide a more precise encoding of meaning than the word-type embeddings. The only exceptions to this trend occurred on the WS353 datasets for the *skip-gram* embeddings, where the word-type embeddings out-performed their paraphrase-level counterparts.

Unsurprisingly, we also find that for the noun benchmarks, the 768-dimensional BERT paraphrase embeddings out-performed the 300-dimensional *skip-gram* representations. However, for two of the three benchmarks containing verbs, the smaller *skip-gram* transfer embeddings achieve higher scores than the larger BERT embeddings. On SimLex-999, which contains a mixture of parts of speech, the BERT embeddings performed better than the *skip-gram* embeddings for nouns, while *skip-gram* out-performed BERT on verbs and adjectives.

In terms of the methods for calculating similarity between available paraphrase embeddings, best results were achieved by the max and shortest path methods. This contradicts an earlier finding by Dubossarsky et al. (2018), who showed that most previous work on multi-sense embeddings reports the best scores achieved using the mean method. They explain that taking the mean similarity is equivalent to sub-sampling and multiple estimation of word vector representations, thereby reducing bias in the cosine similarity calculations. To examine these results more closely, we compare the paraphrase embeddings chosen by the max and shortest path methods for different word pair comparisons.

Word Pair	Human Rating	Shortest Path	Maximum Similarity
mud, dirt	7.3	$v_{mud \rightarrow dirt}$, $v_{dirt \rightarrow mud}$	$v_{mud \rightarrow earth}$, $v_{dirt \rightarrow sand}$
bar, jail	1.9	$v_{bar \rightarrow court}$, $v_{jail \rightarrow custody}$	$v_{bar \rightarrow club}$, $v_{jail \rightarrow lock-up}$
plead, beg	9.1	$v_{plead \rightarrow beg}$, $v_{beg \rightarrow plead}$	$v_{plead \rightarrow beg}$, $v_{beg \rightarrow urge}$
multiply, divide	1.8	$v_{multiply \rightarrow spread}$, $v_{divide \rightarrow spread}$	$v_{multiply \rightarrow strengthen}$, $v_{divide \rightarrow subdivide}$

Table 24: Paraphrase embeddings selected by shortest path and maximum similarity methods to represent word pairs from SIMLEX-999.

In summary, we have proposed a method for using the paraphrase-specific contexts present in PSTS to generate term representations at the sub-word level based on two different embedding techniques. Through evaluation on word similarity and relatedness prediction benchmarks, we demonstrate that these paraphrase embeddings capture meaning more precisely than their word-type level counterparts.

6.3. Applications 2: Word Sense Induction

PSTS provides sentence-level contexts for different senses of a target word. The second application setting we use to evaluate the utility of PSTS is in using PSTS' sense-specific contexts to aid in the task of word sense induction (WSI).

Word sense induction is the task of discriminating the different meanings, or senses, of a target word that are present in some corpus (see Section 2.3.1). Systems are presented with a set of sentences containing a shared target word. Each sentence must be annotated with a sense label, such that sentences where the target has the same meaning get the same label. Importantly, unlike in the related task of word sense disambiguation, systems are not provided with a pre-defined sense inventory to guide the labeling of the different senses. Systems must both determine how many senses exist for each target, and properly assign the same label to each same-sense instance. Systems are evaluated by comparing the predicted labeling to a human-annotated set of 'ground truth' sense labels for each sentence.

Our approach to WSI incorporates both the paraphrase sense clusters produced in Chapter 3, and the paraphrase-level embeddings produced from PSTS in Section 6.2.3. We assume that the sense clusters for a target word represent its possible meanings, and use the paraphrase embeddings as a bridge to map each target word instance to the most appropriate sense cluster. Note that while our WSI model operates very much like a WSD model in that it maps word instances to senses from an underlying sense inventory, an important distinction is that we assume no prior knowledge of the sense inventory (WordNet) used for evaluation. Instead, we produce our own sense inventory in an unsupervised way through paraphrase clustering.

6.3.1. WSI Method

Specifically, given a target word t, we call its set of PPDB sense clusters $C = \{c_1, c_1, \ldots, c_k\}$. Each sense cluster c_i contains a set of paraphrases of t: $c_i = \{p_1, p_2, \ldots, p_m\}$ (such that for each p_j, $t \leftrightarrow p_j$ is a paraphrase in PPDB). Each paraphrase has an associated paraphrase embedding that represents its shared sense with t, $v_{t \to p}$. The task presents our system with a set of target word instances, $s_1, s_2, \ldots, s_n$. Each is a short passage of text containing the target t. We denote as v_s a vector embedding that represents the context of the target t in sentence s. In order to map each target word instance s to the most appropriate sense cluster c, we compare the context representation v_s to the set of paraphrase representations in c, $V_c = \{v_{t \to p} : p \in c\}$,[5] via an affinity function $f(v_s, V_c)$. For example, a target instance for the target word $t =$*plant* might be the sentence $s =$*The $\underline{\underline{\text{plant}}}$ employs between 800 and 900 on three shifts*, and the word *plant* in this context would be represented using a vector v_s. This instance can be compared to the PPDB sense cluster for *plant*, $c = \{\text{station}, \text{powerplant}\}$, by calculating the value of an affinity function that takes the context vector v_s and paraphrase vectors $v_{plant \to station}$ and $v_{plant \to powerplant}$ as input. Figure 28 depicts this process.

We experiment with two affinity functions, average ($f_{avg}(v_s, V_c)$) and maximum ($f_{max}(v_s, V_c)$):

$$f_{avg}(v_s, V_c) = \underset{p \in c}{\text{avg}} \cos(v_s, v_{t \to p}) \tag{6.4}$$

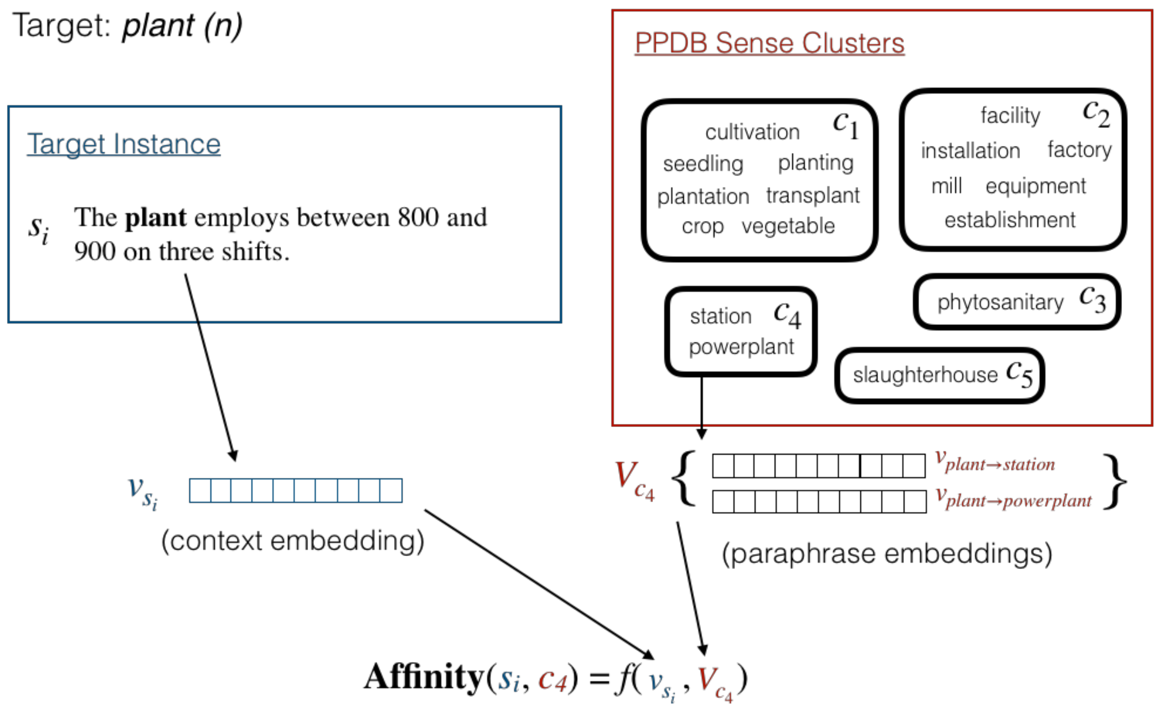
$$f_{max}(v_s, V_c) = \max_{p \in c} \cos(v_s, v_{t \to p}) \tag{6.5}$$

Each comparison function takes in a contextual embedding from a target word instance, and a set of paraphrase embeddings from a sense cluster, and produces a score that indicates the affinity between the target word instance and the sense cluster. We assign each instance s to the cluster which maximizes the comparison function.

6.3.2. Experiments

The datasets used for our WSI experiment come from two shared tasks – SemEval-2007 Task 2 (Agirre and Soroa, 2007) and SemEval-2010 Task 14 (Manandhar et al., 2010).

[5]In practice, we experiment with using embeddings in both directions: *target* embeddings $v_{t \to p}$, and *paraphrase* embeddings $v_{p \to t}$, and report the results for both settings. For the rest of the method description we just use notation for the target direction for brevity.



Target: *plant (n)*
Target Instance
s_i The **plant** employs between 800 and 900 on three shifts.
PPDB Sense Clusters
cultivation seedling planting plantation transplant crop vegetable c_1
facility installation factory mill equipment establishment c_2
phytosanitary c_3
station powerplant c_4
slaughterhouse c_5
v_{s_i}
(context embedding)
V_{c_4}
$v_{plant \to station}$
$v_{plant \to powerplant}$
(paraphrase embeddings)
Affinity$(s_i, c_4) = f(v_{s_i}, V_{c_4})$

Figure 28: Illustration of process for calculating the affinity between a target instance of *plant (n)* (s_i) and a PPDB sense cluster (c_4). The context embedding for the target instance (v_{s_i}) is compared to the (*plant* $\leftrightarrow$ *) embeddings for paraphrases in c_4. The target instance will be assigned to the sense cluster which maximizes the affinity function f, which may be one of f_{avg} (Eq. 6.4) or f_{max} (Eq. 6.5).

SemEval-2007 contains 27,312 sentences for 100 target nouns and verbs, and SemEval-2010 contains 8,915 sentences for 100 noun and verb targets. In both cases, the ground truth sense annotations are derived from WordNet 1.7.1 senses. The targets in SemEval-2007 have 3.68 senses on average, and there are 3.85 senses on average for targets in SemEval-2010. Clustering quality metrics are used to evaluate system output for each target word, by comparing clusters formed by sentences with the same predicted sense to clusters formed by sentences with the same ground truth sense.

In addition to experimenting with the function used to map a target word instance to a sense cluster, we also vary our experiments along two additional axes: the type of contextual representation used to represent target word instances (each type associated with a particular flavor of paraphrase embedding), and the direction of the paraphrase embeddings (target $v_{t \to p}$ vs. paraphrase $v_{p \to t}$). The contextual representations used are:

- **BERT**: To represent target t in sentence s, we use the 768-dimensional contextualized embedding for t generated by the same pre-trained BERT model used in Section 6.2.2. The complementary paraphrase embeddings used in this setting are PP-BERT.
- **SG-WIN5**: In this setting, we represent the context of target t in sentence s by averaging the skip-gram context embeddings from words appearing within a window of 5 words to either side of t.[6] The skip-gram model used is the same used to initialize the PP-SG embeddings, and the complementary paraphrase embeddings used in this setting are PP-SG.

Our WSI method is applied to the SemEval-2007 and SemEval-2010 datasets, varying (a) the function used to map sentences to clusters (f_{avg} vs. f_{max}), (b) the type of contextual representation used (BERT vs SG-WIN5), and (c) the direction of paraphrase embedding used (paraphrase $v_{p \to t}$ vs. target $v_{t \to p}$). As the assumed sense inventory, we use PPDB sense clusters generated using our best-performing spectral method from Chapter 3,[7] where clus-

[6]We also experimented with window widths of 1 and 3, but 5 out-performed them in all experiments.

[7]This spectral method uses PPDBSCORE to measure the similarity between paraphrases to be clustered,

ters for each target are formed by clustering all the target's paraphrases having PPDBSCORE at least 2.3.

We also implement several baselines for comparison:

- **KMeans**. For each type of context embedding, we run KMeans clustering on the contexts for sentences to be labeled, setting k to the number of PPDB sense clusters.
- **Most Frequent Sense (MFS)**. The most-frequent-sense baseline assigns the same sense label to all sentences for a given target.
- **Random**. This baseline executes 10 random clusterings for each target, with the number of clusters set to the number of ground truth senses. We report average scores over the 10 runs.

6.3.3. Results

The predicted mappings produced by each method are compared to the ground truth sets of human-annotated WordNet 1.7.1 senses. Results for each dataset are reported in Tables 25 and 26. Table 25 reports the results in terms of paired F-Score, and Table 26 reports the results in terms of adjusted rand index (ARI), a metric that does not share the positive bias toward the most-frequent-sense baseline (i.e. assigning all sentences for a target word to a single sense). Appendix A.1 provides more details on each of these evaluation metrics.

For both datasets and context embedding types, we find that our method of assigning a sense to a target word instance by mapping its context embedding to a PPDB sense cluster via paraphrase embeddings out-performs the baseline of K-means clustering on the context embeddings alone. Moreover, our method's performance would have placed it second in both shared tasks among all original task participants in terms of paired F-score, and first in Sem-Eval 2010 overall in terms of ARI by a substantial margin.

and monolingual contextual similarity (operationalized as the cosine similarity of 300-dimensional *skip-gram* word embeddings trained on the Google News corpus) as input to the silhouette score metric used to determine the optimal number of senses.

Ctx. Embedding	WSI Method	SemEval-2007	Sem-Eval 2010
BERT	Kmeans	0.484	0.414
	f_{avg}, Target PP-BERT	0.604	0.567
	f_{max}, Target PP-BERT	0.641	0.588
	f_{avg}, Paraphrase PP-BERT	0.672	**0.623**
	f_{max}, Paraphrase PP-BERT	0.680	0.578
SG-WIN5	Kmeans	0.466	0.439
	f_{avg}, Target PP-SG	0.609	0.529
	f_{max}, Target PP-SG	**0.685**	0.545
	f_{avg}, Paraphrase PP-SG	0.613	0.532
	f_{max}, Paraphrase PP-SG	0.677	0.546
	MFS Baseline	**0.789**	**0.635**
	Random Baseline	0.379	0.319
	Best in Task	0.787*	0.633**

Table 25: WSI results in terms of paired F-Score. Numbers reported are the weighted average FScore over the 100 targets in each dataset, where each target is weighted by the number of applicable sentences. Our systems' best output would have ranked them as 2nd among participants in both competitions, behind the top-scoring systems *UBC-AS (SemEval-2007) and **Duluth-WSI-SVD-Gap (SemEval2010).

One somewhat surprising result is that representing the context of each target word instance using its 768-dimensional contextualized BERT token embedding did not consistently outperform the method of averaging 300-dimensional *skip-gram* embeddings within a context window. This indicates that although BERT token embeddings do encode information about the context of each token via attention mechanisms within the Transformer encoder architecture (Vaswani et al., 2017), they do not capture enough of this contextual information for us to ignore the surrounding tokens entirely for context-sensitive tasks.

In summary, this set of WSI experiments indicates that micro-sense embeddings derived from PSTS can be used in conjunction with PPDB sense clusters to discriminate and label target word instances with their specific meaning in context.

6.4. Applications 3: Contextual Hypernym Prediction

Finally, we aim to demonstrate that PSTS can be used to automatically construct a dataset for a contextual lexical semantic prediction task, *without* the need for any human annota-

Ctx. Embedding	WSI Method	SemEval-2007	Sem-Eval 2010
BERT	Kmeans	0.004	0.006
	f_{avg}, Target PP-BERT	-0.004	**0.218**
	f_{max}, Target PP-BERT	0.000	0.150
	f_{avg}, Paraphrase PP-BERT	-0.003	0.163
	f_{max}, Paraphrase PP-BERT	0.005	0.122
SG-WIN5	Kmeans	0.003	0.004
	f_{avg}, Target PP-SG	-0.001	0.080
	f_{max}, Target PP-SG	0.008	0.067
	f_{avg}, Paraphrase PP-SG	-0.003	0.077
	f_{max}, Paraphrase PP-SG	**0.012**	0.070
MFS Baseline		0.000	0.000
Random Baseline		0.000	0.002
Best in Task		0.022*	0.043**

Table 26: WSI results in terms of adjusted rand index (ARI). Numbers reported are the weighted average ARI over the 100 targets in each dataset, where each target is weighted by the number of applicable sentences. Our systems' best results would have placed 2nd and 1st among participants in the 2007 and 2010 competitions respectively. Top-scoring systems in each competition in terms of ARI were *UPV.SI (SemEval-2007) and **Duluth-WSI-CO-PK2 (SemEval2010).

tion, pre-defined sense inventory, or pre-trained word sense disambiguation model. The task used as the testbed for demonstration is predicting hypernymy in context.

Most previous work on hypernym prediction has been done out of context. In this setting, the input to the task is a pair of terms like (*table*, *furniture*), and the model aims to predict whether the second term is a hypernym of the first (in this case, it is). However, more recently, both Shwartz and Dagan (2016a) and Vyas and Carpuat (2017) have pointed out that hypernymy between two terms depends on the contexts in which they appear. Consider the following sentences:

He set the glass down on the $\underline{\underline{\text{table}}}$.

Results are reported in $\underline{\underline{\text{table}}}$ 3.1.

She entertained the $\underline{\underline{\text{table}}}$ with her jokes.

In the first context, the $\underline{\text{table}}$ in question is indeed a type of *furniture*. However, in the second and third, the term *table* is used with different meanings, and in these cases is not a hyponym of *furniture*. This is the motivation for studying the task of predicting hypernymy within a given context, where the input to the problem is a pair of sentences each containing a target word, and the task is to predict whether a hypernym relationship holds between the two targets. Example task instances are given in Table 27.

Ex.	Target Word	Related Word	Contexts	Hypernym?
(a)	chessboard	board	The bottom **chessboard** is the realm of cross-border transactions that occur outside of government control. With such an unequal position on the **board**, any efforts to seek a draw are pathetic when the council is about to checkmate us.	Yes
(b)	day	night	Legislation should change attitudes, although change could not occur from one **day** to the next. The **night** before you put very pertinent questions to the parents.	No
(c)	fiberboard	board	The fluting or corrugated **fiberboard** shall be firmly glued to the facings. Industrial plants produce paper and **board** with a capacity exceeding 20 tons per day.	Yes
(d)	chessboard	board	The bottom **chessboard** is the realm of cross-border transactions that occur outside of government control. These people are already on **board** fishing vessels and we should use them to maximum advantage to understand the characteristics of those fisheries.	No

Table 27: Examples of target and related words that may be hypernyms in some sense, depending on the contexts in which they appear.

Previous work on this task has relied on either human annotation, or the existence of a manually-constructed lexical semantic resource (i.e. WordNet), to generate training data. In the case of Shwartz and Dagan (2016a), who examined fine-grained entailment relations in context, a dataset of 3,750 sentence pairs was compiled by automatically extracting sentences from Wikipedia containing target words of interest, and asking crowd workers to manually label sentence pairs with the appropriate fine-grained entailment relation. Subsequently, Vyas and Carpuat (2017) studied the related task of hypernym prediction in context.[8] They generated a larger dataset of 22k sentence pairs which used example sentences from WordNet as contexts, and WordNet's ontological structure to find sentence pairs where the presence or absence of a hypernym relationship could be inferred. This section builds on both previous works, in that we generate an even larger dataset of 116k sentence pairs for studying hypernymy in context, and use the existing test sets for evaluation. However, unlike the previous methods, our dataset is constructed without any manual annotation or reliance on WordNet for contextual examples. Instead, we leverage the sense-specific contexts in PSTS to generate sentence pairs automatically.

6.4.1. Producing a Hypernym Prediction Training Set

Because PSTS can be used to query sentences containing target words with a particular fine-grained sense, our hypothesis is that, given a set of term pairs with known semantic relations, we can use PSTS to automatically produce a large, high-quality training set of sentence pairs for contextual hypernym prediction. More generally, our goal is to generate training instances of the form:

$$(t, w, c_t, c_w, l)$$

where t is a target term, w is a possibly related term, c_t and c_w are contexts, or sentences, containing t and w respectively, and l is a binary label indicating whether t and w are a hyponym-hypernym pair in the senses as they are expressed in contexts c_t and c_w. The proposed method for generating such instances from PSTS relies on WordNet (or another lexical semantic resource) only insofar as we use it to enumerate term pairs (t, w) with known semantic relation; the contexts (c_t, c_w) in which these relations hold or do not are

[8]Fine-grained entailment prediction and hypernym prediction are closely related; in an upward-monotone sentence, a hyponym entails its hypernym, e.g. *virus* entails *bug* in *"I caught a stomach $\underline{\text{virus}}$."*

generated automatically from PSTS.

The training set is deliberately constructed to contain instances representing each of the following desired types:

(a) Positive instances, where (t, w) hold a hypernym relationship in contexts c_t and c_w $(l = 1)$ (Table 27, examples a and c).

(b) Negative instances, where (t, w) hold some semantic relation other than hypernymy (such as meronymy or antonymy) in contexts c_t and c_w $(l = 0)$. This will encourage the model to discriminate true hypernym pairs from other semantically related pairs (Table 27, example b).

(c) Negative instances, where (t, w) hold a known semantic relation, including possibly hypernymy, in some sense, but the contexts c_t and c_w are not indicative of this relation $(l = 0)$. This will encourage the model to take context into account when making a prediction (Table 27, example d).

Beginning with a target word t, the procedure for generating training instances of each type from PSTS is as follows:

- **Find related terms.** The first step is to find related terms w such that the pair (t, w) are related in WordNet with relation type r (which could be one of synonym, antonym, hypernym, hyponym, meronym, or holonym), and $t \leftrightarrow w$ is a paraphrase pair present in PSTS. The related terms are not constrained to be hypernyms, in order to enable generation of instances of type (b) above.

- **Generate contextually related instances** (types (a) and (b) above). Given term pair (t, w) with known relation r, generate sentence pairs where this relation is assumed to hold as follows. First, order PSTS sentences in $S^{\dot{t}w}$ (containing target t) and $S^{t\dot{w}}$ (containing related term w in its sense as a paraphrase of t) by decreasing quality score, as predicted by the regression model from Section 5.5.3. Next, choose the top-k sentences from each ordered list, and select sentence pairs $(c_t, c_w) \in S^{\dot{t}w} \times S^{t\dot{w}}$ where both sentences are in their respective top-k lists. Add each sentence pair to the dataset as a positive instance ($l = 1$) if $r =$ hypernym, or as a negative instance ($l = 0$) if r is something other than the hypernym relation.

- **Generate contextually unrelated instances** (type (c) above). Given term pair (t, w) with known relation r, generate sentence pairs where this relation is assumed *not* to hold as follows. First, pick a confounding term w' that is a paraphrase of w (i.e. $w \leftrightarrow w'$ is in PPDB), but unrelated to the target t in PPDB. This confounding term is designed to represent an alternative sense of w. In order to select a confounding term that is most different in meaning from the target, choose the paraphrase of w whose word embedding (based on some word embedding model) has lowest cosine similarity with the embedding of t. Next, select the top-$k/2$ sentences containing related term w in its sense as w' from $S^{\dot{w}w'}$ in terms of quality score. Combine these sentences c_w with sentences c_t drawn from the top-k sentences from $S^{\dot{t}w}$ in the previous step to form negative instances. Repeat the process in the other direction, choosing a confounding term t' corresponding to an alternative sense of t, and combine sentences from $S^{\dot{t}t'} \times S^{t\dot{w}}$ to form additional negative instances.

To form the contextual hypernym prediction dataset, this process is carried out over a set of 3,558 target nouns drawn from the Shwartz and Dagan (2016a) and Vyas and Carpuat (2017) datasets, as well as nouns within the top-10k most frequent words in the Google ngrams corpus (after throwing away the first 1k words as stop words). For each target noun, all hypernyms, hyponyms, synonyms, antonyms, co-hyponyms, and meronyms from WordNet were selected as related terms. The number of sentences, k, selected for each target/related term pair was 3. This process resulted in a dataset of 116k instances, of which 28% are positive contextual hypernym pairs (type (a)). The 72% of negative pairs are made up of 34% instances where t and w hold some relation other than hypernymy in context (type (b)), and 38% instances where t and w are unrelated in the given context.

6.4.2. Predicting Hypernymy in Context

Having automatically generated a dataset from PSTS for studying hypernymy in context, the next steps are to adopt a contextual hypernym prediction model to train on the dataset, and then to evaluate its performance on existing hypernym prediction test sets.

The model adopted for predicting hypernymy in context is a fine-tuned version of the BERT pre-trained transformer model (Devlin et al., 2019) (Figure 29). Specifically, we use BERT in its configuration for sentence pair classification tasks, where the input consists of two tokenized sentences (c_t and c_w), preceded by a '`[CLS]`' token and separated by a '`[SEP]`' token. In order to highlight the target t and related term w in each respective sentence, we surround them with left and right bracket tokens "$<$" and "$>$". The model predicts whether the sentence pair contains contextualized hypernyms or not by processing the input through a transformer encoder, and feeding the output representation of the '`[CLS]`' token through fully connected and softmax layers.

6.4.3. Experiments

To test our hypothesis that PSTS can be used to generate a large, high-quality dataset for training a contextualized hypernym prediction model, we perform experiments that compare the performance of the BERT hypernym prediction model on existing test sets after training on our PSTS dataset, versus training on only the original, or the combined original and PSTS, training sets.

There are two existing datasets for contextual hypernym prediction that are used in our experiments. The first, which we abbreviate as S&D-binary, is a binarized version of the fine-grained entailment relation dataset from Shwartz and Dagan (2016a). While the original dataset contained five different entailment types, we convert all *forward-entailment* and flipped *reverse-entailment* instances to positive (hypernym) instances, and the rest to negative instances. The resulting dataset has 3750 instances (18% positive and 82% negative), split into train/dev/test portions of 2630/190/930 instances respectively. The second

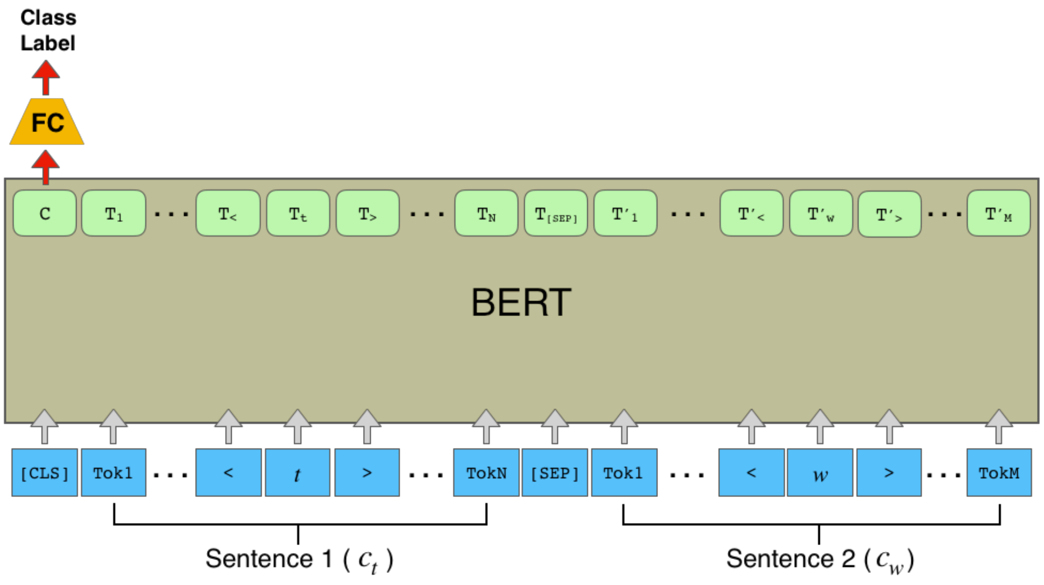


Figure 29: Illustration of the contextual hypernym prediction model based on fine-tuning BERT (Devlin et al., 2019). Input sentences c_t and c_w are tokenized, prepended with a `[CLS]` token, and separated with a `[SEP]` token. The target word t in the first sentence, c_t, and the related word w in the second sentence, c_w, are highlighted by surrounding them with `<` and `>` tokens. The class label (*hypernym* or *not*) is predicted by feeding the output representation of the `[CLS]` token through fully-connected and softmax layers.

dataset used in our experiments is "WordNet Hypernyms in Context" (WHiC) from Vyas and Carpuat (2017). It contains 22,781 instances (23% positive and 77% negative), split into train/dev/test portions of 15716/1704/5361 instances respectively.

For both datasets, we compare results of the BERT sentence pair classification model on the test portions after fine-tuning on the PSTS dataset alone, the original training set alone, or a combination of the PSTS dataset with the original training set. In order to gauge how similar the datasets are to one another, we also experiment with training on S&D-binary and testing on WHiC, and vice versa. In each case we use the dataset's original dev portion for tuning the BERT model parameters (batch size, number of epochs, and learning rate).

6.4.4. Results

Results are reported in terms of weighted average F-Score over the positive and negative classes, and given in Table 28.

Training Set	**Test Set**	**F1**
S&D-binary	WHiC	0.686
WHiC		**0.787**
PSTS		0.722
PSTS+WHiC		0.783
S&D-binary	S&D-binary	0.792
WHiC		0.717
PSTS		0.803
PSTS+S&D-binary		**0.833**

Table 28: Performance of the BERT fine-tuned contextual hypernym prediction model on two existing test sets, segmented by training set. All results are reported in terms of weighted average F1.

In the case of S&D-binary, we find that training on the 116k-instance PSTS dataset leads to a modest improvement in test set performance of 1.4% over training on the original 2.6k-instance training set. Combining the PSTS and original training sets leads to a more substantial 5.2% performance over training on the original dataset alone. However, on the WHiC dataset, it turns out that training on the PSTS dataset as opposed to the original 15.7k-instance training set leads to a relative 8.5% drop in performance. The WHiC results obtained by the BERT classifier after training on the original dataset are equivalent to the best results reported in Vyas and Carpuat (2017) – 0.54 F1 for the positive (hypernym) class.

Training on S&D-binary/testing on WHiC and vice versa gives the lowest scores for both datasets, indicating that there is something characteristically different between the two datasets. Given that training with PSTS improves performance on S&D-binary but not on WHiC suggests that PSTS is more similar to S&D-binary.

In conclusion, our experiments indicate that the sense-specific contexts in PSTS can be used to automatically generate a large dataset for training a contextual hypernym classifier that leads to better performance than training on a small dataset of hand-annotated instances (S&D-binary), and nearly comparable performance to training on a dataset generated from a hand-crafted resource (WHiC). This suggests that it is worth exploring the use of PSTS to generate sense-specific datasets for other contextual lexical semantic tasks.

6.5. Conclusion

This chapter aimed to demonstrate the utility of PSTS via three downstream tasks. The first task was to train paraphrase-level embeddings, which capture word meaning at a fine-grained level. We showed via semantic similarity and relatedness benchmarks that these sub-word-level embeddings captured a more precise notion of semantic similarity than their word type-level counterparts. Next, we demonstrated how to use the sense-specific instances of target words in PSTS within a system for word sense induction (WSI), by using the sentences as a bridge to map WSI test instances in context to their most likely sense cluster (as produced in Chapter 3). Finally, we leveraged PSTS to automatically produce a training set for the task of contextualized hypernym prediction, without the need for a sense tagging model, manual annotation, or existing hand-crafted lexical semantic resources. To evaluate this training set, we adopted a hypernym prediction model based on the BERT transformer (Devlin et al., 2019), and showed that this model, when trained on the large PSTS training set, produces more accurate in-context hypernym predictions than the same model trained on a small hand-crafted training set.

The work in this chapter and the previous supports the primary assertion of this thesis that bilingually-induced paraphrases provide useful signals for computational modeling of lexical semantics – in this case, for modeling fine-grained word sense. Because the paraphrase set for a target word contains terms pertaining to its various senses, we can view paraphrases as instantiating the possible fine-grained senses of a word. Using the pivot method it is possible to automatically extract usages of each target word that pertain to each of its paraphrases. These example usages can then be viewed as a (micro-) sense tagged corpus, and used for training sense-aware models via distributional methods.

CHAPTER 7 : Conclusion

The ability to model the meanings of words and their inter-relationships is key to the long-standing goal of natural language understanding. By and large, the bulk of work in computational modeling of lexical semantics has been focused on learning from signals present in large monolingual corpora – including the distributional properties of words and phrases, and the lexical and syntactic patterns within which they appear. Each of these signals, while useful, has its own drawbacks related particularly to challenges in modeling polysemy or coverage limitations. The goal of this thesis has been to examine bilingually-induced paraphrases as a different source of signal for learning about the meanings of words and their relationships. The key characteristics of such paraphrases that make them well-suited to the task are their wide (and noisy) scope, their natural coverage of both words and phrases, and the inclusion of multiple meanings among the paraphrases of a polysemous target word. The previous chapters explored how paraphrases from the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013; Pavlick et al., 2015b) can be exploited to model word sense, predict scalar adjective intensity, and generate sense-specific examples of word usage. In doing so, it was shown that these key characteristics of paraphrases complement the weaknesses of other monolingual signals. Combining paraphrase-based information with these other signals leads to better models of lexical semantics.

7.1. Summary of Contributions

The first half of this thesis focused on models that directly incorporate features derived from bilingually-induced paraphrases for lexical semantic tasks, beginning with a study of word sense. One of the key characteristics of paraphrases that make them useful for studying word sense is that the set of paraphrases for a polysemous target word contains terms pertaining to each of its various senses (Apidianaki et al., 2014). Whereas traditional approaches to word sense induction have focused on clustering the *contexts* within which a polysemous word appears to uncover its senses, we took the related approach of clustering

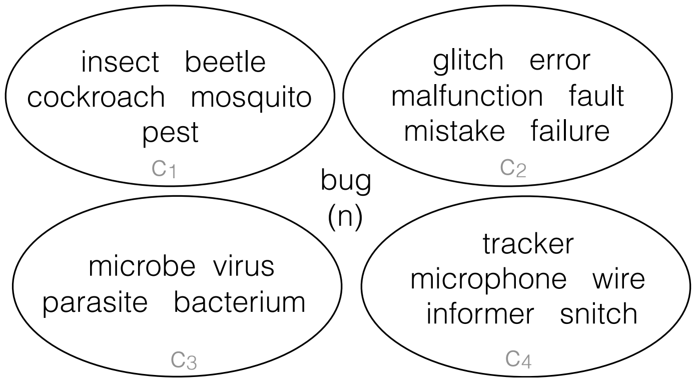

insect beetle cockroach mosquito pest
c_1
glitch error malfunction fault mistake failure
c_2
bug (n)
microbe virus parasite bacterium
c_3
tracker microphone wire informer snitch
c_4

Figure 30: Repeated from Section 3.7, this figure depicts our goal in Chapter 3 to partition paraphrases of an input word like *bug* into clusters representing its distinct senses.

a polysemous word's *paraphrases* in order to enumerate its different meanings.

In Chapter 3, we presented a systematic study of various methods for clustering paraphrases by word sense (Figure 30). Not only did we leverage a word's paraphrases to represent its various senses, but we also examined the second-order relationships that exist between terms within the paraphrase set that can be used to delineate those senses. Our experimental setup compared two clustering algorithms utilizing five different measurements of inter-word similarity, including paraphrase strength, monolingual distributional similarity, and overlapping translations. Because the number of senses for a word is unknown, we also proposed a method for automatically choosing the optimal number of sense clusters based on the Silhouette Coefficient (Rousseeuw, 1987) cluster quality metric. By evaluating clustering output against two sets of ground truth sense clusters, it was shown that using paraphrase strength as a method for computing inter-word similarity produced consistently high-quality clusters, regardless of the clustering algorithm used. However, the best overall results were achieved by combining paraphrase strength and monolingual distributional similarity as metrics for measuring intra-word similarity and selecting the optimal number of clusters, showing that these two signals are complementary to one another.

Our sense clustering study in Chapter 3 was followed by demonstration of how to apply the sense clusters to the downstream task of lexical substitution (lexsub) – suggesting a ranked list of meaning-preserving substitutes for a target word in context. We proposed the 'sense promotion' method as a post-processing step to improve the precision of lexsub models that are based on neural word embeddings. Sense promotion works by elevating the rank of a model's predicted substitutes that belong to the target's most appropriate sense cluster in the given context. Using sense clusters generated in the first half of the chapter in this setting led to a 19% improvement in average precision-at-5 for a state-of-the-art embedding-based lexsub model when evaluated over a test set of approximately 2000 target word instances.

Next, in Chapter 4, we shifted focus to using paraphrase-based signals in the task of predicting relative scalar adjective intensity. The adjectives *funny* and *hilarious* both describe humor, but *funny* is less intense than *hilarious.* The goal of our model was to predict the relative intensity relationship between a pair of such scalar adjectives describing a common attribute. Here, as in the previous chapter, we developed a model that directly incorporated features derived from bilingually-induced paraphrases, and compared the performance of that model to models derived from lexico-syntactic patterns and a manually-compiled adjective intensity lexicon. The paraphrase-based features were extracted from over 36k adjectival phrase paraphrase pairs under the assumption that, for example, paraphrase pair *seriously funny* $\leftrightarrow$ *hilarious* suggests that *funny* $<$ *hilarious.* Due to the wide coverage and noisiness of PPDB, the paraphrase-based model could make predictions for more adjective pairs than could the pattern-based or lexicon-based models, but with lower accuracy. By evaluating on the downstream tasks of globally ranking sets of scalar adjectives by intensity and inferring the polarity of indirect answers to yes/no questions, we showed that combining the wide-coverage paraphrase-based model with the more precise pattern- and lexicon-based models led to better performance on both tasks over using any single model in isolation.

The second half of this thesis further explored the relationship between a target word's paraphrases and its senses. One perennial challenge in distributional models of semantics is the issue of polysemy: a given word type can have (sometimes drastically) different mean-

particularly pleased	$\leftrightarrow$	*ecstatic*
quite limited	$\leftrightarrow$	*restricted*
rather odd	$\leftrightarrow$	*crazy*
so silly	$\leftrightarrow$	*dumb*
completely mad	$\leftrightarrow$	*crazy*

Figure 31: In Chapter 4, we used paraphrases from PPDB of the form $RB\ JJ_u \leftrightarrow JJ_v$ to infer pairwise intensity relationships ($JJ_u < JJ_v$).

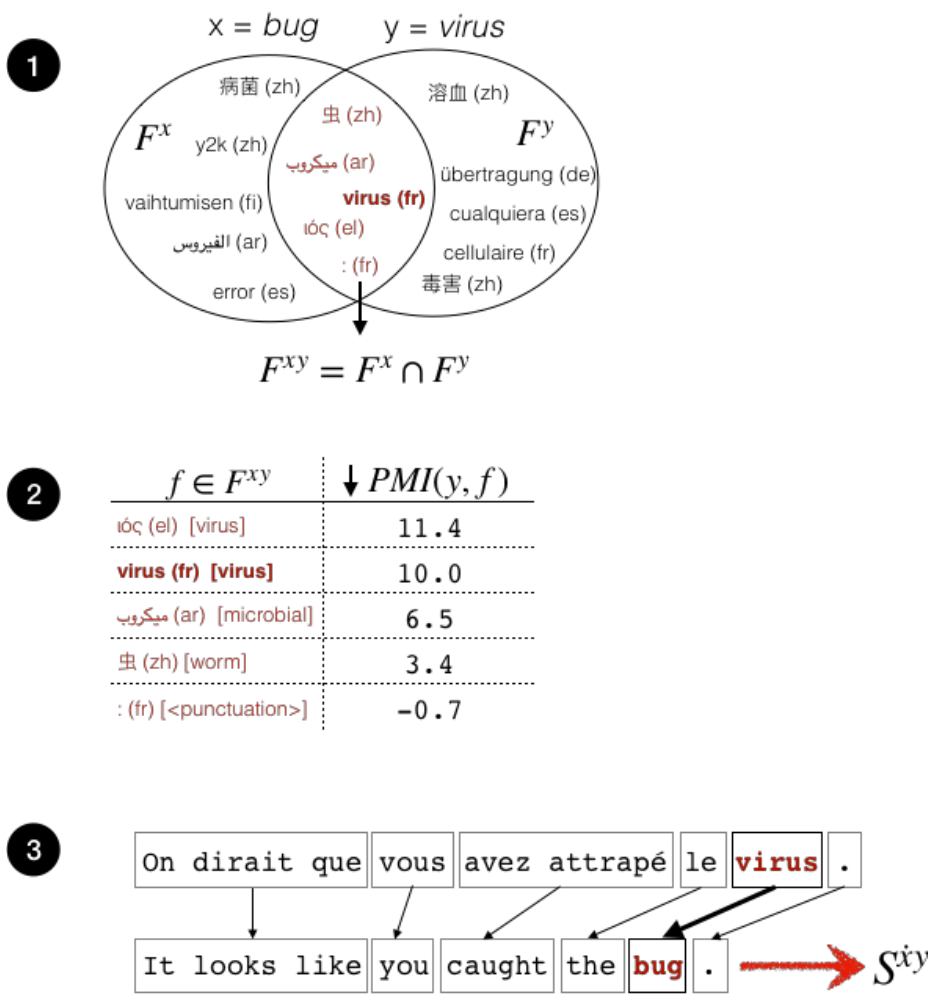
ings depending on its context. Any attempt to represent meaning at the word type level, therefore, confounds a word's different senses in a single type-level representation. For many tasks that rely on modeling word meaning within a particular context, such as recognizing textual entailment, this type-level representation is insufficient. However, it is challenging to construct training corpora for these tasks where words must be used in a particular sense. Researchers building sense-aware corpora typically resort to manual annotation (Edmonds and Cotton, 2001; Mihalcea et al., 2004; Hovy et al., 2006), crowdsourcing (Huang et al., 2012; Shwartz and Dagan, 2016a), or rely on existing manually-compiled lexical semantic resources (Vyas and Carpuat, 2017). Paraphrases can help.

In Chapter 5, we used bilingually-induced paraphrases to extract sentences that are indicative of a particular sense of a polysemous word. Our proposed method exploits the idea that paraphrases for a target word represent its various meanings, coupled with the ability to extract paraphrase instances at scale through bilingual pivoting. Unlike some previous methods for producing sense-tagged corpora, ours does not rely on manual annotation or having a pre-trained word sense disambiguation model. The resulting collection of sentences, which is called Paraphrase-Sense-Tagged Sentences (PSTS), contains up to 10k sentence-level contexts for more than 3M paraphrases in PPDB. The sentences for each paraphrase pair are characteristic of the shared meaning of that pair. For example, sentences for the paraphrase pair *hot* $\leftrightarrow$ *spicy* include "*People should shun $\underline{\underline{\text{hot}}}$ dishes*," while sentences for the paraphrase *hot* $\leftrightarrow$ *popular* include "*This area of technology is $\underline{\underline{\text{hot}}}$.*" We evaluated the quality of sentences in PSTS with the assistance of crowd workers, who indicated that the majority of sentences for a paraphrase pair were indeed indicative of that pair's meaning. We then used the crowd annotations to train a sentence ranking model, which assigns high scores to the sentences for a paraphrase pair that are most characteristic of the pair's meaning.

Chapter 6 demonstrated how to use PSTS as a training bed for lexical semantic models that must incorporate word sense. First, based on the extreme assertion that a word has as many micro-senses as it has paraphrases, we used PSTS as a corpus for training sub-word (paraphrase-level) embeddings based on monolingual distributional models of word representation. Evaluating these paraphrase embeddings on a variety of semantic similarity and relatedness benchmarks, we showed that they out-perform their word type-level embedding counterparts. Next, we applied these paraphrase embeddings to a word sense induction (WSI) task. In this case, the paraphrase embeddings were used as a bridge to map target word instances to their most likely sense cluster (as induced in Chapter 4). This method produced competitive scores on test sets from two previous SemEval WSI shared tasks. Finally, we used PSTS to automatically produce a large training set (116k instances) for the task of predicting hypernymy in context, without the need for manual annotation or reliance on WordNet. To assess the quality of the training set, we adopted a hypernym prediction model based on the BERT transformer encoder (Devlin et al., 2019), and showed that this model, when trained on the PSTS training set, out-performed the same model trained on a manually-labeled training set by 5% relative improvement in F-Score.

7.2. Evolving Models of Word Sense

A recurring theme throughout this thesis has been the tension between discrete versus continuous notions of word sense. Chapter 3 assumes that there exists a discrete partitioning of paraphrases for a target word into sense clusters, whereas Chapters 5-6 throw away this assumption and instead use paraphrases to represent the fine-grained meanings of a target word. As the chapters are laid out chronologically, it is worth mentioning the rationale behind this shift in sense modeling from discrete to fine-grained.



$f \in F^{xy}$	$\downarrow PMI(y, f)$
ιός (el) [virus]	11.4
virus (fr) [virus]	10.0
ميكروب (ar) [microbial]	6.5
虫 (zh) [worm]	3.4
: (fr) [<punctuation>]	-0.7

Figure 32: In Chapter 5, we extracted sentences containing the noun $x =$*bug* in its $y =$*virus* sense from parallel corpora for PSTS by (1) finding translations shared by *bug* adn *virus*, (2) ranking the translations to prioritze *bug*'s translations most 'characteristic' of its meaning in the *virus* sense, and (3) extracting sentences where *bug* was aligned to highly-ranked French translations from bitext corpora.

In Chapter 3 we took a simplified view that word senses can be discretely partitioned by clustering paraphrases. We assumed that for each target word, there exists a set of disjoint senses, and that these senses can be represented by a human-generated partitioning of paraphrases (e.g. Figure 30). The goal of our automatic clustering method was to replicate the human-generated paraphrase clusters as closely as possible. While we briefly acknowledged that varying degrees of sense granularity may be better suited to different tasks, we adopted an intrinsic cluster quality metric to choose an 'optimal' number of senses for each word.

There are two primary issues with the assumption of a 'ground truth' sense inventory adopted in Chapter 3. First, humans have notoriously low agreement in manual sense-tagging tasks (Cinková et al., 2012). In our work, we noted low agreement in the related tasks of crowd clustering (Appendix A.2) and later the evaluation of sentence-paraphrase quality in PSTS (Section 5.5.3). Second, the granularity of sense distinctions that matter can vary depending on the situation or application. For example, in Figure 30, *bug*'s paraphrases *virus* and *bacterium* are clustered together because they are both micro-organisms that can make people sick. If someone is warned, *"Wash your hands often – there's a bug going around."* the given sense inventory is sufficient; hand washing can prevent the spread of both types. However, for a clinician, the distinction between *virus* and *bacterium* is all-important because it impacts how the disease should be treated. Sense distinctions that matter can vary, based on the situational context.

After completing the work in Chapter 3, our initial intent was to combine sense clustering with hypernym prediction in order to develop a new method for taxonomy induction. The method for generating meaning-specific examples of word usage in Chapter 5 was originally conceived as a way to generate sentence-level contexts for each sense cluster, in order to make contextualized hypernym predictions. Ultimately, the taxonomy-building effort was frustrated by the issues of low human agreement and situation-dependent sense distinctions noted above.

However, in building PSTS, we realized that its abstraction of one-paraphrase-per-meaning was a more generalizable approach to word sense modeling than sense clustering. PSTS abandons the assumption of a single ground-truth sense inventory for each target word – although if a user prefers to map each paraphrase to some underlying sense inventory, it is straightforward to do so (as we did during the WSI experiment in Chapter 6). Interestingly, we found that this fine-grained, yet still discrete, model of word sense could be more useful than a completely continuous model in some settings; during the WSI experiment in Chapter 6, clustering the continuous BERT embeddings for target word instances using K-Means did not perform well as a baseline for mapping word instances to a sense inventory, while mapping target word instances to sense clusters via paraphrase embeddings did. This indicates that BERT, which is a state-of-the-art model for text representation that has excellent performance in many language understanding benchmarks, still does not capture all we need to know about word sense.

The main question remains – which method for sense modeling is best? Our answer is that it depends; both the coarse-grained, discrete representation in Chapter 3 and the fine-grained, paraphrase-based representation in Chapter 5 can be useful insofar as they help improve performance on some downstream task. The former was shown to be helpful for lexical substitution, the latter was useful for building precise multi-sense embeddings and a contextualized hypernym prediction dataset, and the two were successfully used in combination for the task of WSI. However, in general, representing fine-grained senses as paraphrases is a more flexible approach that avoids the rigid assumption of an underlying sense inventory.

7.3. Discussion and Future Work

The most important conclusion of this thesis is that signals from bilingually-induced paraphrases can be effectively used within computational models of lexical semantics. Furthermore, when used in combination with signals from monolingual corpora like word distribution and lexico-syntactic patterns, paraphrases provide complementary information that leads to more robust models. One reason is that the paraphrase set for a target word covers many of the target word's possible meanings, and therefore paraphrases can be used to model word sense as we saw in Chapters 3 and 5-6. Another important characteristic of paraphrases is that because the pivot method used to extract them is derived from phrase-based machine translation, paraphrases naturally contain multi-word phrases – not just single words. We saw how adjectival phrase paraphrases could be leveraged to generate features that indicate relative adjective intensity in Chapter 4. Finally, because paraphrases can be extracted automatically and at scale, their wide coverage complements the limited coverage of other signals like lexico-syntactic patterns and manually-compiled lexicons, which was demonstrated in Chapter 4.

This thesis leaves open several questions, which represent limitations of this study and may be areas for future research. First, we have limited our study to paraphrases induced bilingually via the pivot method in PPDB. In no place do we compare with paraphrases automatically generated using other methods, such as monolingual distributional techniques (Lin and Pantel, 2001b,a), monolingual machine translation (Quirk et al., 2004), or neural back-translation (Iyyer et al., 2018). Therefore it is unclear whether our conclusions can be extended to paraphrases in general, or if they are limited to PPDB paraphrases. Second, our study of using paraphrases for generating sense-tagged corpora in Chapters 5 and 6 does not compare directly with other methods for sense tagging, such as supervised word sense disambiguation models. While it may be possible to argue that the additional noise introduced by using our unsupervised method instead of supervised methods is a worthwhile tradeoff because no pre-training is needed, we cannot argue this definitively without directly comparing both methods in order to quantify the differences in accuracy.

One natural extension of this work would be the application of paraphrase-based signals to other problems in lexical semantics. These studies should be focused on problems that could benefit from the strengths of paraphrases, such as those that require awareness of word sense, can benefit from comparison of multi-word phrases to their single-word equivalents, or need high-coverage features. One example of such a task might include taxonomy induction (e.g. Snow et al. (2006); Kozareva and Hovy (2010); Ustalov et al. (2017), and others), where it has been shown that explicitly modeling word sense enables the use of more efficient algorithms (Cocos et al., 2018a). Another possible extension is the application of our method for extracting features from PPDB for relative adjective intensity prediction to other semantic relationships; for example, it may be possible to apply a similar technique to predicting hypernymy (i.e. *small dog* $\leftrightarrow$ *puppy* implies a puppy is a type of dog).

Another question is how to best apply the structured lexical semantic models such as those that are output by our work (e.g. sense clusters and adjective scales) to downstream tasks. An important trend over the past several years in natural language processing has been the shift toward building end-to-end neural models for language understanding tasks such as question answering, sentiment prediction, and natural language inference. These models, while powerful, largely lack the ability to make general inference about facts and relationships that are not explicitly mentioned in their training text. For example, the BERT model, which achieves human-level performance on the SQUAD extractive question answering task where the answer to a question must be located within a span of text (Devlin et al., 2019; Rajpurkar et al., 2016), falls short on the more difficult ARC challenge set of grade school multiple choice science questions where inference over external facts is required (Clark et al., 2018)[1]. An exciting line of future work is building end-to-end models that can reference and reason over structured semantic resources. There is some research in this general area already for reasoning over knowledge bases (e.g. Khashabi et al. (2016); Xiong et al. (2017)) that provides a strong starting point. Expanding this work to deal with multiple sources of information, and resolve uncertainty and noise in the knowledge resources, would enable us to integrate structured lexical semantic resources like those produced within this thesis into powerful end-to-end models for language understanding.

[1]`https://leaderboard.allenai.org/arc/submissions/`, accessed 05 Jan 2019

APPENDIX

A.1. Evaluation Metrics

This appendix provides further detail on evaluation metrics that are used throughout the thesis.

A.1.1. Classification Metrics

Assume a binary classification task, where each item within a set has a ground truth class label (either ***p**ositive* or ***n**egative*), and a predicted class label (also ***p′**ositive* or ***n′**egative*). The items can be partitioned into subsets based on their true and predicted classes:

		Actual Class	
		p	**n**
Pred. Class	**p′**	True Positive (TP)	False Positive (FP)
	n′	False Negative (FN)	True Negative (TN)

Precision Precision measures the ratio of true positives (TP) to all predicted positives ($TP \cup FP$):

$$precision = \frac{|TP|}{|TP \cup FP|}$$

That is, precision estimates the likelihood that an item predicted to have the positive class label is actually positive.

Recall Recall measures the ratio of true positives (TP) to actual positives ($TP \cup FN$):

$$recall = \frac{|TP|}{|TP \cup FN|}$$

That is, recall estimates the share of truly positive items that have been classified as positive.

F-Score F-Score is the harmonic mean of precision and recall:

$$fscore = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

A.1.2. Cluster Comparison Metrics

Cluster comparison metrics are designed to quantify the quality of a predicted clustering by comparing it to a set of ground truth or 'reference' clusters.

Given a set of items to be clustered of size N, let $C = \{c_i | i = 1 \dots n\}$ be a partition of the N items into n reference classes, and $K = \{k_j | j = 1 \dots m\}$ be a partition of the N items into m predicted clusters. A contingency table, recording the assignment of each item to a reference class i and predicted cluster j, is given by $A = \{a_{ij}\}$, where each a_{ij} is the number of items from reference class c_i that have been assigned to predicted cluster k_j, that is $a_{ij} = |c_i \cap k_j|$.

Paired F-Score Frames the clustering problem as a classification task (Manandhar et al., 2010). It first generates the set of all pairs of items belonging to the same reference cluster, $F(C)$. The number of such pairs is given by $|F(C)| = \sum_{i=1}^{|C|} \binom{|c_i|}{2}$. It then generates the set of all pairs of items belonging to the same predicted cluster, $F(K)$. The number of such pairs is given by $|F(K)| = \sum_{j=1}^{|K|} \binom{|k_j|}{2}$.

Precision, recall, and F-score can then be calculated in the usual way, i.e. $precision =$

$\frac{F(K) \cap F(C)}{F(K)}$, $recall = \frac{F(K) \cap F(C)}{F(C)}$, and $fscore = \frac{2 \cdot precision \cdot recall}{precision + recall}$.

Note that when the predicted clustering assigns all items to the same cluster (the *most frequent sense* baseline), the recall is equal to 1. In general, Paired F-Score is known to give a high score to the MFS baseline, and to be biased toward giving high scores to clustering solutions with a small quantity of large predicted clusters.

V-Measure assesses the quality of the clustering solution against reference clusters in terms of clustering homogeneity and completeness (Rosenberg and Hirschberg, 2007).

Homogeneity describes the extent to which each cluster k_j is composed of paraphrases belonging to the same reference class c_i. It is defined by the conditional entropy of the class distribution given the predicted clustering, $H(C|K)$. A clustering is perfectly homogeneous ($H(C|K) = 0$) when each predicted cluster contains only items from the same reference class. In the case where there is only one reference class (and thus $H(C) = 0$), homogeneity is defined to be 1.

$$homg. = \begin{cases} 1 & \text{if } H(C) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & \text{otherwise} \end{cases}$$

$$H(C|K) = -\sum_{k=1}^{|K|} \sum_{c=1}^{|C|} \frac{a_{ck}}{N} \log \frac{a_{ck}}{\sum_{c=1}^{|C|} a_{ck}}$$

$$H(C) = -\sum_{c=1}^{|C|} \frac{\sum_{k=1}^{|K|} a_{ck}}{N} \log \frac{\sum_{k=1}^{|K|} a_{ck}}{N}$$

Completeness refers to the extent to which all points in a reference cluster c_i are captured in a single predicted cluster k_j. It is defined by the conditional entropy of the predicted clustering given the class distribution, $H(K|C)$. A clustering is perfectly complete ($H(K|C) = 0$) when each predicted cluster contains all items from a single reference class.

$$comp. = \begin{cases} 1 & \text{if } H(K) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & \text{otherwise} \end{cases}$$

$$H(K|C) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|} \frac{a_{ck}}{N} \log \frac{a_{ck}}{\sum_{k=1}^{|K|} a_{ck}}$$

$$H(K) = -\sum_{k=1}^{|K|} \frac{\sum_{c=1}^{|C|} a_{ck}}{N} \log \frac{\sum_{c=1}^{|C|} a_{ck}}{N}$$

V-Measure is the harmonic mean of homogeneity and completeness:

$$\text{V-Measure} = \frac{2 \cdot homg. \cdot comp.}{homg. + comp.}$$

Note that in the case that the predicted clustering assigns each item to its own singleton class (sometimes referred to as the *one-cluster-per-item* baseline), the homogeneity is equal to 1. Thus the V-Measure is high for this baseline. In general, V-Measure is known to be biased toward giving high scores to predicted clusterings having a large number of small clusters.

Adjusted Rand Index (ARI) The Rand Index (RI) computes the similarity between a clustering solution and reference clusters by considering all possible pairs of clustered elements, and comparing pair assignment (to same or different clusters) in the reference to the pairs' assignments in the clustering solution (Hubert and Arabie, 1985). Specifically, if a gives the number of pairs of items that are assigned to the same cluster and have the same reference class, and b gives the number of pairs of items that are assigned to different clusters and have different reference classes, then the Rand Index is computed as:

$$RI = \frac{a+b}{\binom{N}{2}}$$

The ARI adjusts the RI for chance (Hubert and Arabie, 1985; Pedregosa et al., 2011), and can be calcluated using the contingency table A:

$$ARI = \frac{\sum_{ck}\binom{a_{ck}}{2} - \left[\sum_{c}\binom{a_{c*}}{2}\sum_{k}\binom{a_{*k}}{2}\right]/\binom{N}{2}}{\frac{1}{2}\left[\sum_{c}\binom{a_{c*}}{2} + \sum_{k}\binom{a_{*k}}{2}\right] - \left[\sum_{c}\binom{a_{c*}}{2}\sum_{k}\binom{a_{*k}}{2}\right]/\binom{N}{2}}$$

A perfect matching between the predicted and reference clusters will yield the maximum ARI score of 1. The ARI metric does not have the biases toward small or large clusters that Paired F-Score and V-Measure have.

In our work we used the implementations of ARI and V-Measure from the Python Scikit-learn package (Pedregosa et al., 2011).

A.1.3. Correlation Metrics

Correlation metrics are designed to measure the similarity of two rankings. For the following explanations, assume a set of n elements $\{x_1, x_2, \ldots, x_n\}$, and two ranking functions σ_1 and σ_2 such that $\sigma_1(x_i)$ gives the rank of element x_i under the first ranking, and $\sigma_2(x_i)$ gives the rank of element x_i under the second ranking.

Kendall's tau-b (τ_b) . This metric computes the rank correlation between the rankings σ_1 and σ_2, incorporating a correction for ties in one or both lists. Values for τ_b range from -1 to 1, with extreme values indicating a perfect negative or positive correlation, and a value of 0 indicating no correlation between the two lists.

The τ_b metric is calculated in terms of the number of concordant and discordant pairs. A pair (x_i, x_j) is said to be *concordant* if x_i and x_j have the same relative ordering under both rankings; that is, either $\sigma_1(x_i) < \sigma_1(x_j)$ and $\sigma_2(x_i) < \sigma_2(x_j)$, or $\sigma_1(x_i) > \sigma_1(x_j)$

and $\sigma_2(x_i) > \sigma_2(x_j)$. A pair (x_i, x_j) is said to be *discordant* if x_i and x_j have different ordering under the two rankings; that is, either $\sigma_1(x_i) < \sigma_1(x_j)$ and $\sigma_2(x_i) > \sigma_2(x_j)$, or $\sigma_1(x_i) > \sigma_1(x_j)$ and $\sigma_2(x_i) < \sigma_2(x_j)$. A pair (x_i, x_j) is *tied* if either $\sigma_1(x_i) = \sigma_1(x_j)$ or $\sigma_2(x_i) = \sigma_2(x_j)$; a tied pair is neither concordant nor discordant.

$$\tau_b = \frac{(\text{number of concordant pairs}) - (\text{number of discordant pairs})}{\sqrt{(N_1)} \cdot \sqrt{(N_2)}}$$

where N_1 gives the number of pairs that are not tied under σ_1, and N_2 gives the number of pairs that are not tied under σ_2.

Spearman's rho (ρ) . Spearman's ρ rank correlation coefficient gives the Pearson's correlation between rankings σ_1 and σ_2:

$$\rho = \frac{\text{cov}(\sigma_1, \sigma_2)}{\text{std}(\sigma_1) \cdot \text{std}(\sigma_2)}$$

where $\text{cov}(\sigma_1, \sigma_2)$ is the covariance of the rankings, and $\text{std}(\sigma)$ is the standard deviation of a ranking.

A.2. Crowd Clustering Task

This Appendix describes the Human Interface Task (HIT) design for clustering paraphrases by word sense in Chapter 3.

We want reasonable sets of sense-clustered paraphrases against which to evaluate our automatic clustering method. Although WordNet synsets are a well-vetted standard, they are insufficient for the task by themselves because of their limited coverage. Using WordNet alone would only allow us to evaluate our method as applied to the 38% of paraphrases for our target word list in PPDB that intersect WordNet. So instead we combine crowdsourcing and manual review to construct a reasonable human-generated set of sense-clustered paraphrases.

Some of the paraphrase sets in our PPDB XXL dataset contain more than 200 phrases, making it unreasonable to ask a single worker to cluster an entire paraphrase set in one sitting. Instead, we take an iterative approach to crowd clustering by asking individual workers to sort a handful of new paraphrases over multiple iterations. Along the way, as workers agree on the placement of words within sense clusters, we add them to a 'crowd-gold' standard. In each iteration, workers can see the most up-to-date crowd gold clustering solution and are asked to sort new, unclustered paraphrases within it.

A.2.1. Iterative Clustering Methodology

Each clustering iteration t includes a *sort* phase in which workers are presented with a list of m unsorted paraphrases $U^t = \{u_1^t, u_2^t ... u_m^t\}$ for a single target word w, and a partial sense clustering solution $C^{t-1} = \{c_1^{t-1}, c_2^{t-1} ... c_k^{t-1}\}$ as generated in previous iterations. The initial round is unseeded, with $C^0 = \emptyset$. Workers are asked to sort all unsorted words u_i^t by adding them to one or more existing clusters $c_{j \leq k}^t$ or new clusters $c_{j>k}^t$. For each target word, n workers sort the same list U^t in each iteration. We add a word u_i^t to the crowd clustering solution C^tif at least $\tau \times n$ workers agree on its placement, where τ is a threshold parameter.

Consolidating Worker Results

When workers add unsorted words to an existing cluster $c_{j \leq k}$, it is easy to assess worker agreement; we can simply count the share of workers who add word u_i to cluster c_j. But when workers add words to a new cluster, we must do additional work to align the j's between workers.

For unsorted words added to new clusters, we consolidate worker placements in iteration t by creating a graph G with a node for each $u_i \in U^t$ added by any worker to a new cluster $c_{j>k}$. We then add weighted edges between each pair of nodes u_i and u_i' in G by counting the number of workers who sorted u_i and u_i' together in some new cluster. Finally we remove edges with weight less than $\tau \times n$ and take the resulting biconnected components as the set of newly added clusters $C^t \setminus C^{t-1}$.

For quality control, we introduce a 'bogus' word that is obviously not a paraphrase of any word in U^t in each round. We ask workers to identify the bogus word and place it in a trash bin. We ignore the results of workers who fail this quality control measure at least 75% of the time.

Merge Phase

We find qualitatively that consolidating clusters based on biconnected components generates overlapping but incomplete clusters after several iterations. So we include a *merge* phase after every third clustering iteration that enables workers to merge clusters from C^{t-1} before sorting new words into C^t. As with the sorting phase, we merge clusters c_{t-1} and c_{t-1}' if at least $\tau \times n$ workers agree that they should be merged.

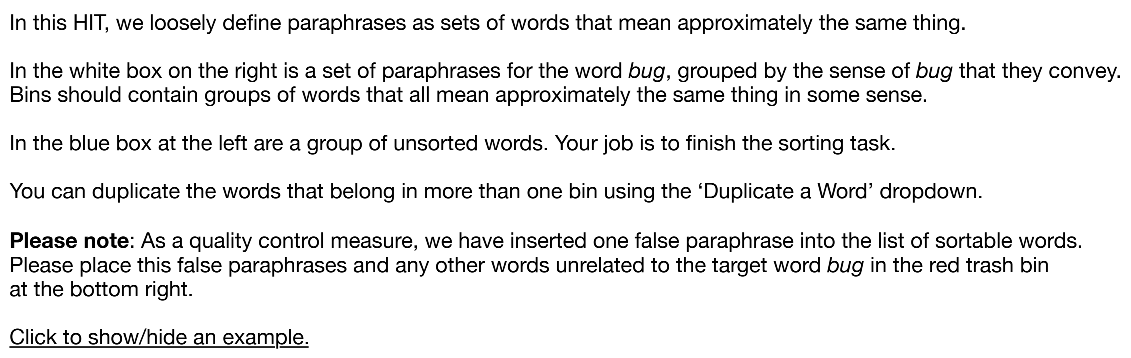
A.2.2. Final Cleanup

Using our method, the size of clusters is monotonically increasing each iteration. So before we use the final crowd-clustered data set, we manually review its contents and make corrections where necessary. Examples of reference clusters used in our experiments are given in Appendix A.3.

A.2.3. User Interface

Our user interface (Figure 33) presents each worker with a 'grab bag' of unclustered words for a given target on the left, and a sorting area on the right. Workers are asked to sort all unclustered words by dragging each one into a bin in the sorting area that contains other words sharing the same sense of the target.

We set the maximum size of the grab bag to be 10 words. This is based on experimentation that showed worker clustering performance declined when the size of the grab bag was larger.


In this HIT, we loosely define paraphrases as sets of words that mean approximately the same thing.

In the white box on the right is a set of paraphrases for the word *bug*, grouped by the sense of *bug* that they convey. Bins should contain groups of words that all mean approximately the same thing in some sense.

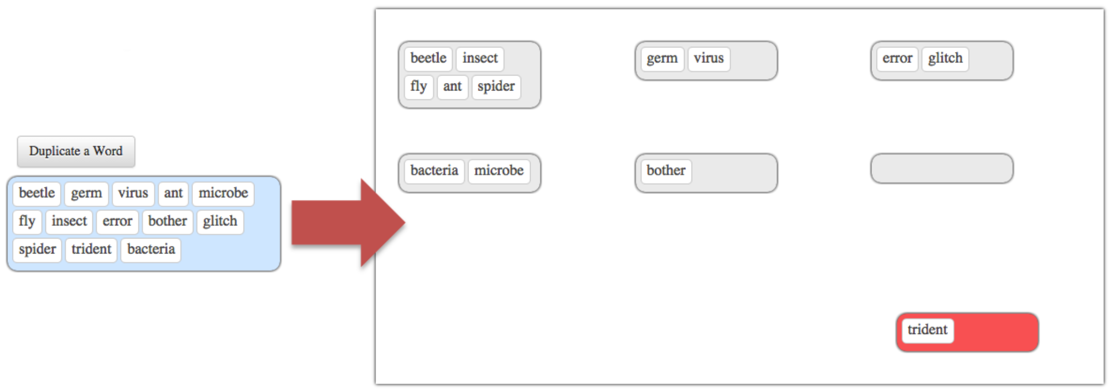
In the blue box at the left are a group of unsorted words. Your job is to finish the sorting task.

You can duplicate the words that belong in more than one bin using the ‘Duplicate a Word’ dropdown.

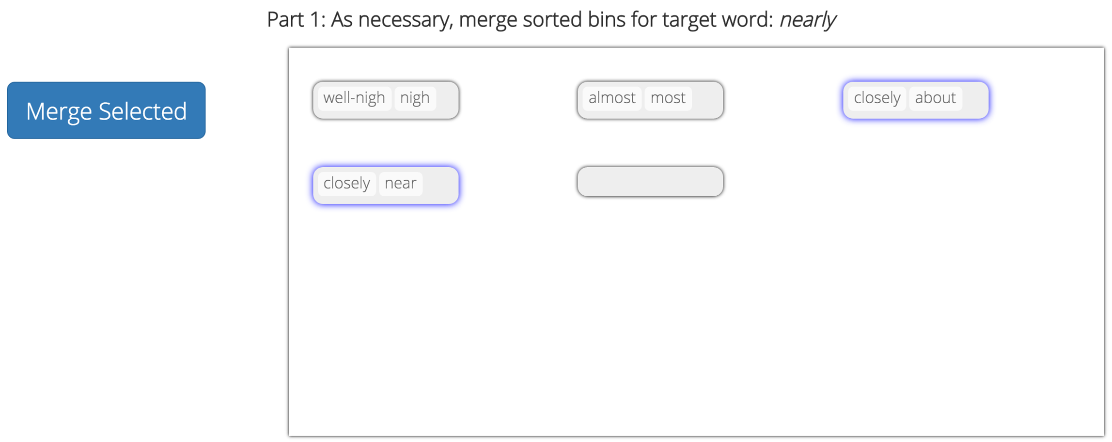
Please note: As a quality control measure, we have inserted one false paraphrase into the list of sortable words. Please place this false paraphrases and any other words unrelated to the target word *bug* in the red trash bin at the bottom right.

Click to show/hide an example.

(a) Sorting user interface instructions to workers.



(b) Sorting user interface.


Part 1: As necessary, merge sorted bins for target word: *nearly*

(c) Merge user interface.

Figure 33: Amazon Mechanical Turk user interface for crowdsourcing reference clusters.

A.3. Example Ground-Truth Clusters

Here we provide examples of ground-truth clusters for the experiments in Chapter 3.

Table 29: WordNet+ Reference Sense Cluster Examples

Query Term	Sense Clusters
film (n)	c_0: wrap, sheet, wrapping
	c_1: flick, picture, telefilm, show, movie, feature, production, documentary
	c_2: episode, sequence, roll, footage, reel, negative, microfilm
	c_3: cinema
touch (v)	c_0: strike, engage, hit, press, feel, handle
	c_1: handle, deal, care
	c_2: strike, affect, move, stir, get
	c_3: be, reach
	c_4: allude, suggest
	c_5: receive, take, have
	c_6: focus on, relate, pertain, regard, concern, involve, apply, affect, hold, refer
	c_7: disturb, modify, violate, change, alter
	c_8: contact, stick, rub, meet, ring, cover
	c_9: impact, hit, influence, bother, modify, alter, treat, strike, affect, stimulate, change
soil (n)	c_0: silt, dirt, subsoil, mud, sand, clay, earth, ground
	c_1: territory
	c_2: farmland, land, sod, bottom, turf, ground, tillage
	c_3: filth, dirt
treat (v)	c_0: feed, provide, cater

Continued…

Table 29: WordNet+ Reference Sense Cluster Examples (continued)

Query Term	Sense Clusters
	c_1: analyze, relieve, analyse, remedy, administer, medicate, nurse, care for, correct, manipulate, operate
	c_2: touch, touch on, run, refine, process, affect, digest
	c_3: react, respond
	c_4: handle, deal, cover, broach, initiate, address, talk about, discuss
	c_5: present, give
	c_6: criminalize, interact, abuse, handle, nurse
severely (r)	c_0: badly, seriously, gravely
	c_1: hard
	c_2: sternly
dark (n)	c_0: nighttime, night
	c_1: shadow, darkness
	c_2: blackness, black, darkness, night
	c_3: darkness
	c_4: darkness

Table 30: CrowdCluster Reference Sense Cluster Examples

Query Term	Sense Clusters
post (n)	c_0: positions, job, occupations, position
	c_1: posting, outpost
	c_2: poste, postal
extended (a)	c_0: extension, extend, expanding, expanded, extending, enlarged, stretched, extensive, expand, increased
	c_1: better, enhanced
	c_2: extending, protracted, stretched, prolonged
let (v)	c_0: continued, remained, retained, had
	c_1: derived, prepared
	c_2: 'm leaving, headed, get going, got to go now, going to do, leaving, leave, be used, got to go
	c_3: shown, saw, showed, demonstrated
	c_4: rented, afforded, hired, rent, rented out, owned
	c_5: dropped, declined
	c_6: forgot, forgotten
	c_7: helped, provided, added, included, offered, gave, awarded
clean (v)	c_0: clean-up, cleanliness, clear, 's clean, get cleaned up, cleansing, wiped clean, cleanse, taken up
	c_1: given up, dropped out
	c_2: is true, potable, drinkable, is healthy, is safe
so (r)	c_0: then, now then, well then, so then
	c_1: yes
	c_2: accordingly, so therefore, therefore, thereby, hence, consequently, thus
	c_3: so too, as well, too

Continued...

Table 30: CrowdCluster Reference Sense Cluster Examples (continued)

Query Term	Sense Clusters
	c_4: very
	c_5: even

A.4. Full Chapter 3 Clustering Results

Full results for all sense clustering experiments from Chapter 3 are given in Tables 31 and 32. The results given in columns $WordNet+$ and $CrowdClusters$ indicate the appropriate metric's weighted average across all query words for that set of reference clusters. The result for each query term is weighted by its number of reference classes.

Table 31: HGFC Clustering Results

SimMethod	Choose K Method	Entailments?	Metric	WordNet+	CrowdClusters
PPDB2.0Score	PPDB2.0Score	False	F-Score	0.3497	0.4571
			V-Measure	0.3906	0.4731
		True	F-Score	0.3504	0.4594
			V-Measure	0.3946	0.4681
	$sim_{PPDB.cos}$	False	F-Score	0.3627	0.4979
			V-Measure	0.3947	0.4797
		True	F-Score	0.3539	0.4897
			V-Measure	0.3929	0.4395
	$sim_{PPDB.js}$	False	F-Score	0.3667	0.4737
			V-Measure	0.3899	0.4346
		True	F-Score	0.3550	0.4969
			V-Measure	0.3896	0.4387
	$sim_{DISTRIB}$	False	F-Score	0.3528	0.4893
			V-Measure	0.3332	0.3755
		True	F-Score	0.3587	0.5095
			V-Measure	0.3375	0.3989
	sim_{TRANS}	False	F-Score	0.3494	0.4336
			V-Measure	0.3571	0.3413
		True	F-Score	0.3562	0.4390
			V-Measure	0.3654	0.3502
$sim_{PPDB.cos}$	PPDB2.0Score	False	F-Score	0.3213	0.5007
			V-Measure	0.3256	0.3198

Continued...

Table 31: HGFC Clustering Results (continued)

SimMethod	Choose K Method	Entailments?	Metric	WordNet+	CrowdClusters
		True	F-Score	0.3465	0.4634
			V-Measure	0.3465	0.4280
	$sim_{PPDB.cos}$	False	F-Score	0.2828	0.4336
			V-Measure	0.4755	0.4569
		True	F-Score	0.3280	0.4425
			V-Measure	0.4548	0.4754
	$sim_{PPDB.js}$	False	F-Score	0.3045	0.4165
			V-Measure	0.4999	0.4622
		True	F-Score	0.3350	0.4691
			V-Measure	0.4187	0.4706
	$sim_{DISTRIB}$	False	F-Score	0.2977	0.4772
			V-Measure	0.3794	0.3270
		True	F-Score	0.3381	0.4422
			V-Measure	0.3662	0.3498
	sim_{TRANS}	False	F-Score	0.3158	0.4102
			V-Measure	0.3373	0.3083
		True	F-Score	0.3276	0.4168
			V-Measure	0.3642	0.3148
$sim_{PPDB.JS}$	PPDB2.0Score	False	F-Score	0.3222	0.4754
			V-Measure	0.3045	0.3482
		True	F-Score	0.3530	0.4570
			V-Measure	0.3703	0.4340
	$sim_{PPDB.cos}$	False	F-Score	0.2839	0.4191
			V-Measure	0.4728	0.4799
		True	F-Score	0.3357	0.4365
			V-Measure	0.4457	0.4595
	$sim_{PPDB.js}$	False	F-Score	0.2952	0.3942
			V-Measure	0.4659	0.4703

Continued...

Table 31: HGFC Clustering Results (continued)

SimMethod	Choose K Method	Entailments?	Metric	WordNet+	CrowdClusters
		True	F-Score	0.3341	0.4452
			V-Measure	0.4391	0.4451
	$sim_{DISTRIB}$	False	F-Score	0.3009	0.4811
			V-Measure	0.3469	0.3535
		True	F-Score	0.3435	0.4781
			V-Measure	0.3563	0.3500
	sim_{TRANS}	False	F-Score	0.3104	0.4026
			V-Measure	0.3114	0.3651
		True	F-Score	0.3247	0.4191
			V-Measure	0.3535	0.3197
$sim_{DISTRIB}$	PPDB2.0Score	False	F-Score	0.2324	0.4476
			V-Measure	0.5261	0.1822
		True	F-Score	0.3311	0.5005
			V-Measure	0.4617	0.4697
	$sim_{PPDB.cos}$	False	F-Score	0.2300	0.4373
			V-Measure	0.5548	0.2467
		True	F-Score	0.3098	0.4920
			V-Measure	0.4724	0.4429
	$sim_{PPDB.js}$	False	F-Score	0.2476	0.4526
			V-Measure	0.4370	0.2681
		True	F-Score	0.3179	0.4847
			V-Measure	0.4935	0.4807
	$sim_{DISTRIB}$	False	F-Score	0.2170	0.3925
			V-Measure	0.5751	0.3977
		True	F-Score	0.2972	0.4663
			V-Measure	0.4905	0.3744
	sim_{TRANS}	False	F-Score	0.2430	0.4036
			V-Measure	0.4942	0.3057

Continued...

Table 31: HGFC Clustering Results (continued)

SimMethod	Choose K Method	Entailments?	Metric	WordNet+	CrowdClusters
		True	F-Score	0.2957	0.4144
			V-Measure	0.4254	0.4056
sim_{TRANS}	PPDB2.0Score	False	F-Score	0.2943	0.4593
			V-Measure	0.2271	0.1530
		True	F-Score	0.3105	0.4587
			V-Measure	0.3094	0.4566
	$sim_{PPDB.cos}$	False	F-Score	0.2969	0.4663
			V-Measure	0.2987	0.2300
		True	F-Score	0.2923	0.4735
			V-Measure	0.3925	0.4353
	$sim_{PPDB.js}$	False	F-Score	0.3027	0.4581
			V-Measure	0.2862	0.1976
		True	F-Score	0.3001	0.4830
			V-Measure	0.3563	0.4340
	$sim_{DISTRIB}$	False	F-Score	0.3001	0.4617
			V-Measure	0.2390	0.2267
		True	F-Score	0.2996	0.4624
			V-Measure	0.3011	0.3367
	sim_{TRANS}	False	F-Score	0.2323	0.3781
			V-Measure	0.4748	0.3106
		True	F-Score	0.2620	0.3887
			V-Measure	0.4095	0.3435

Table 32: Spectral Clustering Results

SimMethod	Choose K Method	Entailments?	Metric	WordNet+	CrowdClusters
PPDB2.0Score	PPDB2.0Score	False	F-Score	0.3268	0.4304
			V-Measure	0.5534	0.5046
		True	F-Score	0.3292	0.4312
			V-Measure	0.5497	0.5326
	$sim_{PPDB.cos}$	False	F-Score	0.3454	0.4865
			V-Measure	0.4698	0.4881
		True	F-Score	0.3517	0.4856
			V-Measure	0.4731	0.4983
	$sim_{PPDB.js}$	False	F-Score	0.3462	0.4858
			V-Measure	0.4556	0.4886
		True	F-Score	0.3510	0.4837
			V-Measure	0.4652	0.4946
	$sim_{DISTRIB}$	False	F-Score	0.3494	0.5067
			V-Measure	0.4452	0.4796
		True	F-Score	0.3570	0.5093
			V-Measure	0.4513	0.4812
	sim_{TRANS}	False	F-Score	0.3231	0.4279
			V-Measure	0.4240	0.4287
		True	F-Score	0.3274	0.4527
			V-Measure	0.4330	0.4330
$sim_{PPDB.cos}$	PPDB2.0Score	False	F-Score	0.3430	0.4888
			V-Measure	0.4823	0.4535
		True	F-Score	0.3317	0.4526
			V-Measure	0.5290	0.4803
	$sim_{PPDB.cos}$	False	F-Score	0.3175	0.4166
			V-Measure	0.5594	0.5244
		True	F-Score	0.3396	0.4635
			V-Measure	0.5019	0.4426

Continued. . .

Table 32: Spectral Clustering Results (continued)

SimMethod	Choose K Method	Entailments?	Metric	WordNet+	CrowdClusters
	$sim_{PPDB.js}$	False	F-Score	0.3176	0.4115
			V-Measure	0.5354	0.5053
		True	F-Score	0.3357	0.4660
			V-Measure	0.4793	0.4265
	$sim_{DISTRIB}$	False	F-Score	0.3381	0.4639
			V-Measure	0.4703	0.5018
		True	F-Score	0.3476	0.4811
			V-Measure	0.4224	0.4115
	sim_{TRANS}	False	F-Score	0.3204	0.4940
			V-Measure	0.4069	0.3706
		True	F-Score	0.3234	0.4437
			V-Measure	0.4089	0.3371
$sim_{PPDB.JS}$	PPDB2.0Score	False	F-Score	0.3389	0.4875
			V-Measure	0.4627	0.4560
		True	F-Score	0.3252	0.4385
			V-Measure	0.5206	0.4753
	$sim_{PPDB.cos}$	False	F-Score	0.3084	0.4109
			V-Measure	0.5442	0.5247
		True	F-Score	0.3327	0.4740
			V-Measure	0.4993	0.4509
	$sim_{PPDB.js}$	False	F-Score	0.3035	0.4003
			V-Measure	0.5233	0.4947
		True	F-Score	0.3327	0.4679
			V-Measure	0.4702	0.4423
	$sim_{DISTRIB}$	False	F-Score	0.3285	0.4701
			V-Measure	0.4581	0.4905
		True	F-Score	0.3412	0.4885
			V-Measure	0.4321	0.4065

Continued...

Table 32: Spectral Clustering Results (continued)

SimMethod	Choose K Method	Entailments?	Metric	WordNet+	CrowdClusters
	sim_{TRANS}	False	F-Score	0.3095	0.4786
			V-Measure	0.3968	0.3385
		True	F-Score	0.3130	0.4550
			V-Measure	0.3955	0.3418
$sim_{DISTRIB}$	PPDB2.0Score	False	F-Score	0.3182	0.5105
			V-Measure	0.4113	0.4587
		True	F-Score	0.3150	0.4454
			V-Measure	0.5241	0.4815
	$sim_{PPDB.cos}$	False	F-Score	0.3160	0.4436
			V-Measure	0.4805	0.5080
		True	F-Score	0.3436	0.4707
			V-Measure	0.4770	0.4574
	$sim_{PPDB.js}$	False	F-Score	0.3124	0.4658
			V-Measure	0.4547	0.5086
		True	F-Score	0.3472	0.4761
			V-Measure	0.4646	0.4313
	$sim_{DISTRIB}$	False	F-Score	0.2813	0.4244
			V-Measure	0.5137	0.5341
		True	F-Score	0.3367	0.4700
			V-Measure	0.4637	0.4465
	sim_{TRANS}	False	F-Score	0.2984	0.4876
			V-Measure	0.3728	0.3685
		True	F-Score	0.3173	0.4501
			V-Measure	0.3876	0.3531
sim_{TRANS}	PPDB2.0Score	False	F-Score	0.2706	0.4461
			V-Measure	0.4154	0.2677
		True	F-Score	0.2617	0.4029
			V-Measure	0.5202	0.4749

Continued. . .

Table 32: Spectral Clustering Results (continued)

SimMethod	Choose K Method	Entailments?	Metric	WordNet+	CrowdClusters
	$sim_{PPDB.cos}$	False	F-Score	0.2636	0.4379
			V-Measure	0.4629	0.3650
		True	F-Score	0.2674	0.4231
			V-Measure	0.5107	0.4268
	$sim_{PPDB.js}$	False	F-Score	0.2647	0.4417
			V-Measure	0.4416	0.3655
		True	F-Score	0.2667	0.4242
			V-Measure	0.5106	0.4250
	$sim_{DISTRIB}$	False	F-Score	0.2652	0.4562
			V-Measure	0.4291	0.3655
		True	F-Score	0.2640	0.4476
			V-Measure	0.5158	0.4111
	sim_{TRANS}	False	F-Score	0.2601	0.4441
			V-Measure	0.4180	0.3240
		True	F-Score	0.2584	0.3850
			V-Measure	0.5131	0.4079

A.5. Crowdsourcing Adjective Scales

In Chapter 4 we utilized two previously-released datasets of gold standard adjective intensity rankings (de Melo and Bansal, 2013; Wilkinson and Oates, 2016), and also generated a third, new set of gold standard adjective scales through crowdsourcing in order to maximize coverage of our JJGraph vocabulary. This appendix details the process of creating the new crowdsourced dataset. Our general approach was, first, to compile *clusters* of adjectives describing a single attribute, and second, to rank adjectives within each cluster by their intensity.

A.5.1. Generating Adjective Sets

We generated clusters of adjectives modifying a shared attribute by partitioning sets of related adjectives associated with a single target word in JJGraph. For example, given the target adjective *hot*, we might generate the following clusters from the set of associated words *warm, heated, boiling, attractive, nice-looking, new*, and *popular*:

$c_1 = \{\text{warm, heated, boiling}\}$
$c_2 = \{\text{attractive, nice-looking}\}$
$c_3 = \{\text{new, popular}\}$

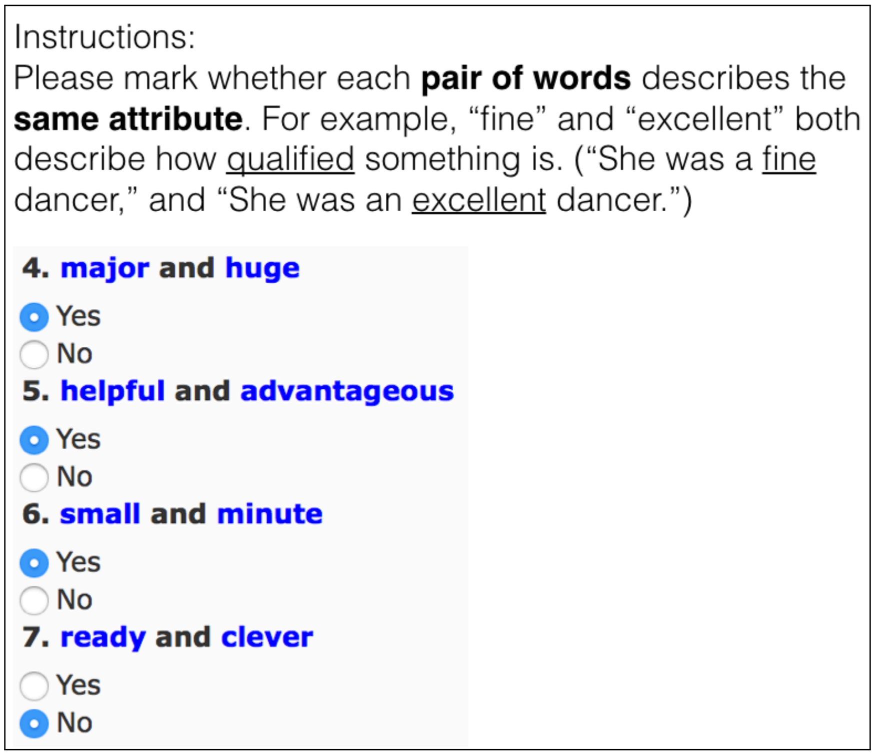
Each cluster represents a sense of the target adjective, and thus the adjectives within a cluster can be ordered along a single scale of increasing intensity. Clusters do not need to be disjoint, as some adjectives have multiple senses.

Partitioning the sets was accomplished with the aid of crowd workers on Amazon Mechanical Turk (MTurk) in two stages. Here we describe the process.

We began by selecting target adjectives with high centrality in JJGraph around which to create gold standard clusters. An adjective has "high centrality" if it is among the 200 most central nodes according to two of three centrality measurements – betweenness centrality, closeness centrality, and degree centrality. With this criterion, we selected 145 target adjectives from JJGraph around which adjective sets were generated.

For each target adjective, we then generated a candidate set of related adjectives to pass to our first MTurk task, which asked workers to remove unrelated adjectives from the candidate sets. We compile an initial candidate set for each of the 145 target adjectives by collecting the first 20 words encountered in a breadth-first search starting at the adjective in JJGraph.

Our first MTurk task aimed to remove unrelated adjectives from the 145 candidate sets (see Figure 34). We presented workers with pairs of adjectives, one being the target adjective and the other a word from that target's candidate set. Three Turkers assessed each pair of adjectives. If a majority of Turkers declared that a pair of adjectives did not describe the same attribute, then the candidate word was removed from that target's set.


Instructions:
Please mark whether each **pair of words** describes the **same attribute**. For example, "fine" and "excellent" both describe how qualified something is. ("She was a fine dancer," and "She was an excellent dancer.")

4. major and huge

[✓ Yes] No

5. helpful and advantageous

[✓ Yes] No

6. small and minute

[✓ Yes] No

7. ready and clever

Yes [✓ No]

Figure 34: First MTurk HIT for constructing gold standard adjective clusters. Each question consists of a target adjective (left) and a cluster candidate adjective (right).

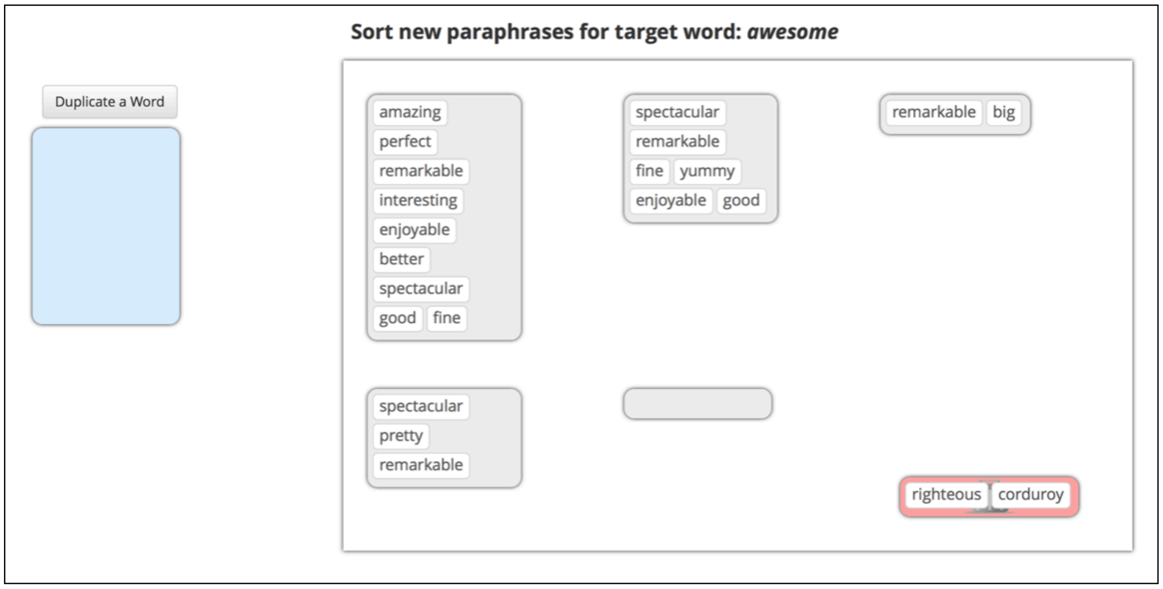


Figure 35: Second MTurk HIT for constructing gold standard adjective clusters.

Once we had a clean set of related adjectives for each target, our second task asked workers to partition the related words (Figure 35). Between 2 and 10 Turkers constructed a clustering for each target adjective. Once a predefined level of agreement was reached among Turkers for a target adjective's clusters, the clusters were deemed "gold."

In total, we constructed gold standard clusterings for 145 adjectives. Each candidate set was partitioned into an average of 3.26 clusters.

A.5.2. Ranking Adjectives in a Cluster

Given a clustering of related adjectives for each of the 145 target words, our next step was to ask MTurk workers to order adjectives within a single cluster by intensity.

We completed the ordering in a pairwise fashion. For each adjective cluster, we asked 3 MTurk workers to evaluate – for each pair of adjectives (j_u, j_v), whether j_u was less, equally, or more intense than j_v. The inter-annotator agreement on this task (Cohen's kappa) was $\kappa = 0.53$, indicating moderate agreement.

Finally, we filtered each cluster to include only adjectives with a unanimous, consistent global ranking. More specifically, if a cluster has adjectives j_u, j_v, and j_w, and workers unanimously agree that $j_u < j_v$ and $j_v < j_w$, then workers must also unanimously agree that $j_u < j_w$ for the ranking to be consistent. After this final step, our dataset consisted of 79 remaining clusters having from 2 to 8 ranked adjectives each (mean 3.18 adjectives per cluster).

A.6. Adapting the Wilkinson Dataset

The Wilkinson dataset (Wilkinson and Oates, 2016) as published provides 12 full adjective scales between polar opposites, e.g. (*ancient*, *old*, *fresh*, *new*). We manually subdivided each scale into half scales for compatibility with the other datasets in this study, producing 21 half scales total. The procedure for dividing a full- into a half-scale was as follows:

1. If the full scale contains two central adjectives where the polarity shifts from negative to positive, sub-divide the scale between them (e.g. divide the scale (*simple*, *easy*, *hard*, *difficult*) between central adjectives *easy* and *hard*).

2. Otherwise, if the full scale contains a central neutral adjective, subdivide the full scale into halves with the neutral adjective belonging to both half scales (e.g. divide (*freezing*, *cold*, *warm*, *hot*) into (*freezing*, *cold*, *warm*) and (*warm*, *hot*)).

3. If any of the resulting half scales has length 1, delete it.

hideous ugly || pretty beautiful gorgeous
dark dim || light bright
same alike similar || ~~different~~
simple easy || hard difficult
parched arid dry || damp moist wet
|| few some several many
horrible terrible awful bad || good great wonderful awesome
freezing cold warm || warm hot
ancient old || fresh new
~~slow~~ || quick fast speedy
miniscule tiny small || big large huge enormous gigantic
idiotic stupid dumb || smart intelligent

Table 33: Converting the 12 Wilkinson full scales to 21 half scales. The || symbol denotes the location where full scales are split into half scales. Strike-through text indicates a half-scale was deleted due to having a single adjective.

Table 33 enumerates the half-scales we generated from the full Wilkinson dataset.

A.7. Full Chapter 4 Results

Only the best results for combined scoring methods were given in the main body of Chapter 4. Here we provide the full results for all combinations attempted on both experiments.

Method	%OOV	Acc.	P	R	F
$score_{\text{socal+pat+pp}}$	0.06	0.642	0.684	0.683	0.684
$score_{\text{socal+pp+pat}}$	0.06	0.642	0.678	0.676	0.677
$score_{\text{socal+pp}}$	0.09	0.634	0.690	0.663	0.676
$score_{\text{socal+pat}}$	0.07	0.634	0.680	0.670	0.675
deMarneffe (2010)	0.02	0.610	0.597	0.594	0.596
$score_{\text{socal}}$	0.26	0.504	0.710	0.481	0.574
$score_{\text{pp+pat}}$	0.06	0.504	0.559	0.547	0.553
$score_{\text{pp+pat+socal}}$	0.06	0.504	0.559	0.547	0.553
$score_{\text{pp+socal+pat}}$	0.06	0.504	0.559	0.547	0.553
$score_{\text{pp+socal}}$	0.09	0.496	0.568	0.533	0.550
$score_{\text{pp}}$	0.09	0.496	0.568	0.533	0.550
$score_{\text{pat+pp}}$	0.06	0.423	0.532	0.517	0.524
$score_{\text{pat+socal+pp}}$	0.06	0.423	0.532	0.517	0.524
$score_{\text{pat+pp+socal}}$	0.06	0.423	0.532	0.517	0.524
$score_{\text{pat+socal}}$	0.07	0.415	0.528	0.504	0.516
$score_{\text{pat}}$	0.07	0.407	0.524	0.491	0.507
all-“YES”	0.00	0.691	0.346	0.500	0.409

Table 34: Full Chapter 4 IQAP Results. Accuracy and macro-averaged precision (P), recall (R), and F1-score (F) over *yes* and *no* responses on 123 question-answer pairs. The percent of pairs having one or both adjectives out of the score vocabulary is listed as %OOV. Rows are sorted by descending F1-score.

		Score Accuracy (before ranking)		Global Ranking Results		
Test Set	Score Type	Cover-age	Pairwise Acc.	Pairwise Acc.	Avg. τ_b	ρ
deMelo	$score_{\text{pat}}$	0.480	0.844	0.650	0.633	0.583
	$score_{\text{pp}}$	0.325	0.458	0.307	0.071	0.09
	$score_{\text{socal}}$	0.277	0.546	0.246	0.110	0.019
	$score_{\text{pat+pp}}$	0.623	0.742	0.619	0.543	0.511
	$score_{\text{socal+pp}}$	0.478	0.523	0.380	0.162	0.106
	$score_{\text{pat+socal}}$	0.609	0.757	0.653	0.609	0.533
	$score_{\text{pat+pp+socal}}$	0.698	0.718	0.637	0.537	0.463
	$score_{\text{pat+socal+pp}}$	0.698	0.722	0.644	0.564	0.482
	$score_{\text{pp+socal+pat}}$	0.698	0.635	0.579	0.393	0.327
	$score_{\text{pp+pat+socal}}$	0.698	0.661	0.599	0.437	0.372
	$score_{\text{socal+pp+pat}}$	0.698	0.647	0.589	0.430	0.341
	$score_{\text{socal+pat+pp}}$	0.698	0.680	0.613	0.496	0.395
Crowd	$score_{\text{pat}}$	0.112	0.784	0.321	0.203	0.221
	$score_{\text{pp}}$	0.738	0.676	0.597	0.437	0.405
	$score_{\text{socal}}$	0.348	0.757	0.421	0.342	0.293
	$score_{\text{pat+pp}}$	0.747	0.696	0.627	0.481	0.432
	$score_{\text{socal+pp}}$	0.812	0.687	0.621	0.470	0.465
	$score_{\text{pat+socal}}$	0.412	0.750	0.476	0.373	0.298
	$score_{\text{pat+pp+socal}}$	0.821	0.686	0.630	0.462	0.440
	$score_{\text{pat+socal+pp}}$	0.821	0.686	0.624	0.465	0.472
	$score_{\text{pp+socal+pat}}$	0.821	0.670	0.630	0.456	0.435
	$score_{\text{pp+pat+socal}}$	0.821	0.670	0.630	0.456	0.435
	$score_{\text{socal+pp+pat}}$	0.821	0.690	0.633	0.481	0.480
	$score_{\text{socal+pat+pp}}$	0.821	0.694	0.639	0.495	0.480
Wilkinson	$score_{\text{pat}}$	0.443	0.852	0.475	0.441	0.435
	$score_{\text{pp}}$	0.795	0.753	0.639	0.419	0.450
	$score_{\text{socal}}$	0.311	0.895	0.312	0.317	0.422
	$score_{\text{pat+pp}}$	0.885	0.833	0.738	0.605	0.564
	$score_{\text{socal+pp}}$	0.795	0.773	0.672	0.484	0.565
	$score_{\text{pat+socal}}$	0.639	0.846	0.59	0.503	0.506
	$score_{\text{pat+pp+socal}}$	0.885	0.833	0.738	0.605	0.564
	$score_{\text{pat+socal+pp}}$	0.885	0.833	0.754	0.638	0.600
	$score_{\text{pp+socal+pat}}$	0.885	0.750	0.672	0.426	0.414
	$score_{\text{pp+pat+socal}}$	0.885	0.750	0.672	0.426	0.414
	$score_{\text{socal+pp+pat}}$	0.885	0.769	0.705	0.492	0.504
	$score_{\text{socal+pat+pp}}$	0.885	0.833	0.754	0.638	0.611

Table 35: Full Chapter 4 pairwise relation prediction and global ranking results.

BIBLIOGRAPHY

E. Agirre and A. Soroa. Semeval-2007 task 02: Evaluating word sense induction and discrimination systems. In Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007), pages 7–12, Prague, Czech Republic, 2007. Association for Computational Linguistics.

E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), pages 19–27, Boulder, Colorado, 2009. Association for Computational Linguistics.

T. W. Anderson and D. A. Darling. A test of goodness of fit. Journal of the American statistical association, 49(268):765–769, 1954.

R. K. Ando. Applying alternating structure optimization to word sense disambiguation. In Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL), pages 77–84, New York, New York, 2006. Association for Computational Linguistics.

M. Apidianaki. Data-driven semantic analysis for multilingual wsd and lexical selection in translation. In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL), pages 77–85, Athens, Greece, 2009a. Association for Computational Linguistics.

M. Apidianaki. Data-Driven Semantic Analysis for Multilingual WSD and Lexical Selection in Translation. In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL), pages 77–85, Athens, Greece, 2009b. Association for Computational Linguistics.

M. Apidianaki. Vector-space models for PPDB paraphrase ranking in context. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2028–2034, Austin, Texas, 2016. Association for Computational Linguistics.

M. Apidianaki and Y. He. An algorithm for cross-lingual sense clustering tested in a MT evaluation setting. In Proceedings of the 7th International Workshop on Spoken Language Translation (IWSLT-10), Paris, France, 2010.

M. Apidianaki, E. Verzeni, and D. McCarthy. Semantic Clustering of Pivot Paraphrases. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC), pages 4270–4275, Reykjavik, Iceland, 2014. European Language Resources Association (ELRA).

R. Banjade, N. Maharjan, N. B. Niraula, V. Rus, and D. Gautam. Lemon and tea are not similar: Measuring word-to-word similarity by combining different methods. In International Conference on Intelligent Text Processing and Computational Linguistics (CICLing), pages 335–346, Cairo, Egypt, 2015. Springer.

C. Bannard and C. Callison-Burch. Paraphrasing with bilingual parallel corpora. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL), pages 597–604, Ann Arbor, Michigan, 2005. Association for Computational Linguistics.

M. Bansal, J. DeNero, and D. Lin. Unsupervised Translation Sense Clustering. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), pages 773–782, Montréal, Canada, 2012. Association for Computational Linguistics.

M. Baroni, R. Bernardi, N.-Q. Do, and C.-c. Shan. Entailment above the word level in distributional semantics. In Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL), pages 23–32, Avignon, France, 2012. Association for Computational Linguistics.

M. Baroni, G. Dinu, and G. 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 (ACL), pages 238–247, Baltimore, Maryland, 2014. Association for Computational Linguistics.

O. Baskaya, E. Sert, V. Cirik, and D. Yuret. Ai-ku: Using substitute vectors and co-occurrence modeling for word sense induction and disambiguation. In Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), pages 300–306, Atlanta, Georgia, 2013. Association for Computational Linguistics.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. Journal of machine learning research, 3(Feb):1137–1155, 2003.

R. Bhagat and E. Hovy. What is a paraphrase? Computational Linguistics, 39(3):463–472, 2013.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. Journal of machine Learning research, 3(Jan):993–1022, 2003.

O. Bodenreider. The unified medical language system (UMLS): integrating biomedical terminology. Nucleic acids research, 32(suppl_1):D267–D270, 2004.

S. Bordag. Word sense induction: Triplet-based clustering and automatic evaluation. In 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL), pages 137–144, Trento, Italy, 2006. Association for Computational Linguistics.

S. Brody and M. Lapata. Bayesian word sense induction. In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL), pages 103–111, Athens, Greece, 2009. Association for Computational Linguistics.

P. F. Brown, S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer. Word-sense disambiguation using statistical methods. In Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL), pages 264–270, Berkeley, California, 1991. Association for Computational Linguistics.

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

E. Bruni, N.-K. Tran, and M. Baroni. Multimodal distributional semantics. Journal of Artificial Intelligence Research, 49:1–47, 2014.

C. Callison-Burch. Syntactic Constraints on Paraphrases Extracted from Parallel Corpora. In Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 196–205, Honolulu, Hawaii, 2008. Association for Computational Linguistics.

M. Carpuat and D. Wu. Improving statistical machine translation using word sense disambiguation. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 61–72, Prague, Czech Republic, 2007. Association for Computational Linguistics.

S. Cederberg and D. Widdows. Using LSA and noun coordination information to improve the precision and recall of automatic hyponymy extraction. In Proceedings of the seventh conference on Natural Language Learning (CoNLL) at HLT-NAACL - Volume 4, pages 111–118, Edmonton, Canada, 2003. Association for Computational Linguistics.

Y. S. Chan and H. T. Ng. Scaling up word sense disambiguation via parallel texts. In Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI), pages 1037–1042, Pittsburgh, Pennsylvania, 2005.

H.-S. Chang, Z. Wang, L. Vilnis, and A. McCallum. Distributional inclusion vector embedding for unsupervised hypernymy detection. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL) - Volume 1 (Long Papers), pages 485–495, New Orleans, Louisiana, 2018. Association for Computational Linguistics.

X. Chen, Z. Liu, and M. Sun. A unified model for word sense representation and disambiguation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1025–1035, Doha, Qatar, 2014. Association for Computational Linguistics.

D. K. Choe and E. Charniak. Naive Bayes word sense induction. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1433–1437, Seattle, Washington, 2013. Association for Computational Linguistics.

S. Cinková, M. Holub, and V. Kríž. Managing uncertainty in semantic tagging. In Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL), pages 840–850, Avignon, France, 2012. Association for Computational Linguistics.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. arXiv preprint arXiv:1803.05457, 2018.

D. Clarke. Context-theoretic semantics for natural language: an overview. In Proceedings of the Workshop on Geometrical Models of Natural Language Semantics (GEMS), pages 112–119, Athens, Greece, 2009. Association for Computational Linguistics.

A. Cocos and C. Callison-Burch. Clustering Paraphrases by Word Sense. In Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pages 1463–1472, San Diego, California, 2016. Association for Computational Linguistics.

A. Cocos, M. Apidianaki, and C. Callison-Burch. Word Sense Filtering Improves Embedding-Based Lexical Substitution. In Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications, pages 110–119, Valencia, Spain, 2017. Association for Computational Linguistics.

A. Cocos, M. Apidianaki, and C. Callison-Burch. Comparing constraints for taxonomic organization. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Volume 1 (Long Papers), pages 323–333, New Orleans, Louisiana, 2018a. Association for Computational Linguistics.

A. Cocos, V. Wharton, E. Pavlick, M. Apidianaki, and C. Callison-Burch. Learning scalar adjective intensity from paraphrases. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1752–1762, Brussels, Belgium, 2018b. Association for Computational Linguistics.

T. Cohn and M. Lapata. Sentence compression beyond word deletion. In Proceedings of the 22nd International Conference on Computational Linguistics (COLING), pages 137–144, Manchester, United Kingdom, 2008. Association for Computational Linguistics.

A. Copestake and T. Briscoe. Semi-productive polysemy and sense extension. Journal of semantics, 12(1):15–67, 1995.

D. A. Cruse. Aspects of the micro-structure of word meanings. Polysemy: Theoretical and computational approaches, pages 30–51, 2000.

I. Dagan. Lexical disambiguation: sources of information and their statistical realization. In Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL), pages 341–342, Berkeley, California, 1991. Association for Computational Linguistics.

I. Dagan and A. Itai. Word sense disambiguation using a second language monolingual corpus. Computational linguistics, 20(4):563–596, 1994.

I. Dagan, O. Glickman, and B. Magnini. The PASCAL recognising textual entailment challenge. In Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising textual entailment, pages 177–190. Springer, 2006.

M.-C. de Marneffe, C. D. Manning, and C. Potts. Was It Good? It Was Provocative. Learning the Meaning of Scalar Adjectives. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL), pages 167–176, Uppsala, Sweden, 2010. Association for Computational Linguistics.

G. de Melo and M. Bansal. Good, Great, Excellent: Global Inference of Semantic Intensities. Transactions of the Association for Computational Linguistics, 1:279–290, 2013.

M. Denkowski and A. Lavie. Meteor-next and the meteor paraphrase tables: Improved evaluation support for five target languages. In Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics (MATR), pages 339–342, Uppsala, Sweden, 2010.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

M. Diab and P. Resnik. An Unsupervised Method for Word Sense Tagging using Parallel Corpora. In Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL), pages 255–262, Philadelphia, Pennsylvania, USA, 2002. Association for Computational Linguistics.

W. Dolan, C. Quirk, and C. Brockett. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In Proceedings of the 20th International Conference of Computational Linguistics (COLING), pages 350–356, Geneva, Switzerland, 2004. COLING.

B. Dorow and D. Widdows. Discovering corpus-specific word senses. In Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics (EACL) - Volume 2, pages 79–82, Budapest, Hungary, 2003. Association for Computational Linguistics.

R. Dror, G. Baumer, S. Shlomov, and R. Reichart. The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), pages 1383–1392, Melbourne, Australia, 2018. Association for Computational Linguistics.

H. Dubossarsky, E. Grossman, and D. Weinshall. Coming to your senses: on controls and evaluation sets in polysemy research. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1732–1740, Brussels, Belgium, 2018. Association for Computational Linguistics.

J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Cybernetics, 3:32–57, 1973.

H. Dyvik. Translations as Semantic Mirrors: from Parallel Corpus to Wordnet. In Proceedings of the ECAI'98 Workshop Multilinguality in the lexicon, pages 24–44, Brighton, UK, 1998.

P. Edmonds and S. Cotton. SENSEVAL-2: overview. In Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems, pages 1–5, Toulouse, France, 2001. Association for Computational Linguistics.

M. Everett and S. P. Borgatti. Ego network betweenness. Social networks, 27(1):31–38, 2005.

C. Fellbaum, editor. WordNet: an electronic lexical database. MIT Press, 1998.

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. ACM Transactions on information systems, 20(1):116–131, 2002.

J. R. Firth. The technique of semantics. Transactions of the philological society, 34(1): 36–73, 1935.

J. R. Firth. A Synopsis of Linguistic Theory 1930-1955. Studies in Linguistic Analysis, pages 1–32, 1957.

R. A. Fisher. The design of experiments. Oliver & Boyd, Edinburgh, 1935.

J. L. Fleiss. Measuring nominal scale agreement among many raters. Psychological bulletin, 76(5):378, 1971.

W. A. Gale, K. W. Church, and D. Yarowsky. Using Bilingual Materials to Develop Word Sense Disambiguation Methods. In Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation, 1992.

J. Ganitkevitch. Large-Scale Paraphrase Extraction and Applications. PhD Thesis, Johns Hopkins University, 2018.

J. Ganitkevitch and C. Callison-Burch. The Multilingual Paraphrase Database. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC), pages 4276–4283, Reykjavik, Iceland, 2014. European Language Resources Association (ELRA).

J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. PPDB: The Paraphrase Database. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pages 758–764, Atlanta, Georgia, 2013. Association for Computational Linguistics.

D. Gerz, I. Vulić, F. Hill, R. Reichart, and A. Korhonen. SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2173–2182, Austin, Texas, 2016. Association for Computational Linguistics.

R. Girju, A. Badulescu, and D. Moldovan. Learning semantic constraints for the automatic discovery of part-whole relations. In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT), pages 80–87, Edmonton, Canada, 2003. Association for Computational Linguistics.

G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. Numerische mathematik, 14(5):403–420, 1970.

N. Green and S. Carberry. Generating indirect answers to yes-no questions. In Proceedings of the Seventh International Workshop on Natural Language Generation, pages 189–198. Association for Computational Linguistics, 1994.

N. Green and S. Carberry. Interpreting and generating indirect answers. Computational Linguistics, 25(3):389–435, 1999.

H. P. Grice. Logic and conversation. 1975, pages 41–58, 1975.

D. Gross and K. J. Miller. Adjectives in wordnet. International Journal of lexicography, 3 (4):265–277, 1990.

J. Guo, W. Che, H. Wang, and T. Liu. Learning sense-specific word embeddings by exploiting bilingual resources. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 497–507, Dublin, Ireland, 2014. Dublin City University and Association for Computational Linguistics.

I. Gurobi Optimization. Gurobi Optimizer Reference Manual. 2016. URL `http://www.gurobi.com`.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. Machine learning, 46(1-3):389–422, 2002.

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

V. Hatzivassiloglou and K. R. McKeown. Towards the Automatic Identification of Adjectival Scales: Clustering Adjectives According to Meaning. In Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL), pages 172–182, Columbus, Ohio, 1993. Association for Computational Linguistics.

M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In Proceedings of the 15th conference on Computational linguistics (COLING) - Volume 2, pages 539–545, Nantes, France, 1992.

F. Hill, R. Reichart, and A. Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. Computational Linguistics, 41(4):665–695, 2015.

J. B. Hirschberg. Scalar implicature and indirect responses to yes-no questions (tech. rep. ms-cis-84-9). University of Pennsylvania, 1984.

J. B. Hirschberg. A theory of scalar implicature (natural languages, pragmatics, inference). University of Pennsylvania Ph.D. PhD Thesis, Thesis, 1985.

D. Hope and B. Keller. Maxmax: a graph-based soft clustering algorithm applied to word sense induction. In Proceedings of the 14th International International Conference on Intelligent Text Processing and Computational Linguistics (CICLing), pages 368–381, Samos, Greece, 2013. Springer.

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. Ontonotes: The 90\% solution. In Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers, New York, New York, 2006. Association for Computational Linguistics.

E. Hovy, Z. Kozareva, and E. Riloff. Toward completeness in concept extraction and classification. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 948–957, Singapore, 2009. Association for Computational Linguistics.

E. Huang, R. Socher, C. Manning, and A. Ng. Improving Word Representations via Global Context and Multiple Word Prototypes. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers), pages 873–882, Jeju Island, Korea, 2012. Association for Computational Linguistics.

L. Hubert and P. Arabie. Comparing partitions. Journal of classification, 2(1):193–218, 1985.

N. Ide and Y. Wilks. Making sense about sense. In Word sense disambiguation, pages 47–73. Springer, 2007.

N. Ide, T. Erjavec, and D. Tufis. Sense discrimination with parallel corpora. In Proceedings of the ACL-02 workshop on Word Sense Disambiguation: Recent Successes and Future Directions, pages 61–66, Philadelphia, Pennsylvania, 2002. Association for Computational Linguistics.

M. Iyyer, J. Wieting, K. Gimpel, and L. Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Volume 1 (Long Papers), pages 1875–1885, New Orleans, Louisiana, 2018. Association for Computational Linguistics.

D. Jurgens and I. Klapaftis. Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), pages 290–299, Atlanta, Georgia, 2013. Association for Computational Linguistics.

H. Kamp and B. Partee. Prototype theory and compositionality. Cognition, 57(2):129–191, 1995.

K. Kawakami and C. Dyer. Learning to represent words in context with multilingual supervision. arXiv preprint arXiv:1511.04623, 2015.

D. Khashabi, T. Khot, A. Sabharwal, P. Clark, O. Etzioni, and D. Roth. Question answering via integer programming over semi-structured knowledge. In Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI), pages 1145–1152, New York, New York, 2016. AAAI Press.

A. Kilgarriff. I don't believe in word senses. Computers and the Humanities, 31(2):91–113, 1997.

A. Kilgarriff. Word senses. In Word Sense Disambiguation, pages 29–46. Springer, 2007.

J.-K. Kim and M.-C. de Marneffe. Deriving adjectival scales from continuous space word representations. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1625–1630, Seattle, Washington, 2013. Association for Computational Linguistics.

I. P. Klapaftis and S. Manandhar. Word sense induction using graphs of collocations. In Proceedings of the 18th European Conference on Artificial Intelligence (ECAI), pages 298–302, Patras, Greece, 2008.

I. P. Klapaftis and S. Manandhar. Word sense induction & disambiguation using hierarchical random graphs. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 745–755, Cambridge, Massachusetts, 2010. Association for Computational Linguistics.

L. Kotlerman, I. Dagan, I. Szpektor, and M. Zhitomirsky-Geffet. Directional distributional similarity for lexical inference. Natural Language Engineering, 16(4):359–389, 2010.

Z. Kozareva and E. 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 (EMNLP), pages 1110–1118, Cambridge, Massachusetts, 2010. Association for Computational Linguistics.

Z. Kozareva, E. Riloff, and E. Hovy. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. Proceedings of ACL-08: HLT, page 1048, 2008.

G. Kremer, K. Erk, S. Padó, and S. 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 (EACL), pages 540–549, Gothenburg, Sweden, 2014.

G. Kruszewski, D. Paperno, and M. Baroni. Deriving boolean structures from distributional vectors. Transactions of the Association for Computational Linguistics (TACL), 3:375–388, 2015.

G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), pages 260–270, San Diego, California, 2016. Association for Computational Linguistics.

E. Lefever, V. Hoste, and M. De Cock. Parasense or how to use parallel corpora for word sense disambiguation. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL): Short Papers-Volume 2, pages 317–322, Portland, Oregon, 2011. Association for Computational Linguistics.

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

B. Levin. English verb classes and alternations: A preliminary investigation. University of Chicago press, 1993.

O. Levy and Y. Goldberg. Dependency-based word embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL) - Volume 2: Short Papers, volume 2, pages 302–308, Baltimore, Maryland, 2014. Association for Computational Linguistics.

O. Levy, S. Remus, C. Biemann, I. Dagan, and I. Ramat-Gan. Do Supervised Distributional Methods Really Learn Lexical Inference Relations? In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pages 970–976, Denver, Colorado, 2015.

J. Li and D. Jurafsky. Do Multi-Sense Embeddings Improve Natural Language Understanding? In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1722–1732, Lisbon, Portugal, 2015. Association for Computational Linguistics.

L. Li, B. Roth, and C. Sporleder. Topic models for word sense disambiguation and token-based idiom detection. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL), pages 1138–1147, Uppsala Sweden, 2010. Association for Computational Linguistics.

X. Li, L. Vilnis, and A. McCallum. Improved Representation Learning for Predicting Commonsense Ontologies. In Proceedings of the ICML 17 Workshop on Deep Structured Prediction, Sydney, Australia, 2017.

D. Lin. Automatic retrieval and clustering of similar words. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL), Volume 2, pages 768–774, Montréal, Canada, 1998. Association for Computational Linguistics.

D. Lin and others. An information-theoretic definition of similarity. In Proceedings of the 15th International Conference on Machine Learning (ICML), volume 98, pages 296–304, Madison, Wisconsin, 1998.

D. Lin and P. Pantel. Dirt@ sbt@ discovery of inference rules from text. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pages 323–328, San Francisco, California, 2001a. ACM.

D. Lin and P. Pantel. Discovery of inference rules for question-answering. Natural Language Engineering, 7(4):343–360, 2001b.

D. Lin and P. Pantel. Discovery of Inference Rules for Question Answering. Natural Language Engineering, 2001c.

D. Lin, S. Zhao, L. Qin, and M. Zhou. Identifying synonyms among distributionally similar words. In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI), volume 3, pages 1492–1493, Acapulco, Mexico, 2003. AAAI Press.

E. Loper and S. Bird. NLTK: The Natural Language Toolkit. In Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1, pages 63–70, Philadelphia, Pennsylvania, 2002. Association for Computational Linguistics.

T. Luong, R. Socher, and C. Manning. Better word representations with recursive neural networks for morphology. In Proceedings of the Seventeenth Conference on Computational Natural Language Learning (CoNLL), pages 104–113, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

B. MacCartney and C. D. Manning. Natural logic for textual inference. In Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing, pages 193–200, Prague, Czech Republic, 2007. Association for Computational Linguistics.

N. Madnani and B. J. Dorr. Generating phrasal and sentential paraphrases: A survey of data-driven methods. Computational Linguistics, 36(3):341–387, 2010.

S. Manandhar, I. Klapaftis, D. Dligach, and S. Pradhan. SemEval-2010 Task 14: Word Sense Induction and Disambiguation. In Proceedings of the 5th International Workshop on Semantic Evaluation(SemEval), pages 63–68, Uppsala, Sweden, 2010. Association for Computational Linguistics.

M. Mancini, J. Camacho-Collados, I. Iacobacci, and R. Navigli. Embedding words and senses together via joint knowledge-enhanced training. In Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL), pages 100–111, Vancouver, Canada, 2017. Association for Computational Linguistics.

M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. Journal of the ACM (JACM), 7(3):216–244, 1960.

D. McCarthy and R. Navigli. SemEval-2007 Task 10: English Lexical Substitution Task. In Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007), pages 48–53, Prague, Czech Republic, 2007. Association for Computational Linguistics.

D. McCarthy and R. Navigli. The English Lexical Substitution Task. Language Resources and Evaluation Special Issue on Computational Semantic Analysis of Language: SemEval-2007 and Beyond, 43(2):139–159, 2009.

D. McCarthy, M. Apidianaki, and K. Erk. Word Sense Clustering and Clusterability. Computational Linguistics, 42(2):245–275, 2016.

O. Melamud, I. Dagan, and J. 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 (NAACL), pages 472–482, Denver, Colorado, 2015a. Association for Computational Linguistics.

O. Melamud, O. Levy, and I. Dagan. A Simple Word Embedding Model for Lexical Substitution. In Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, pages 1–7, Denver, Colorado, 2015b. Association for Computational Linguistics.

O. Melamud, J. Goldberger, and I. Dagan. context2vec: Learning generic context embedding with bidirectional lstm. In Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning (CONLL), pages 51–61, Berlin, Germany, 2016. Association for Computational Linguistics.

R. Mihalcea, T. Chklovski, and A. Kilgarriff. The senseval-3 english lexical sample task. In Proceedings of Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, pages 25–28, Barcelona, Spain, 2004. Association for Computational Linguistics.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. CoRR, abs/1301.3781, 2013a. URL `http://arxiv.org/abs/1301.3781`.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In Advances in Neural Information Processing Systems 26, Lake Tahoe, 2013b.

G. A. Miller. WordNet: A Lexical Database for English. Commun. ACM, 38(11):39–41, Nov. 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL `http://doi.acm.org/10.1145/219717.219748`.

G. A. Miller and W. G. Charles. Contextual correlates of semantic similarity. Language and cognitive processes, 6(1):1–28, 1991.

G. A. Miller, M. Chodorow, S. Landes, C. Leacock, and R. G. Thomas. Using a semantic concordance for sense identification. In HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994, pages 240–243, Plainsboro, New Jersey, 1994. Association for Computational Linguistics.

M. Morzycki. Modification. Cambridge University Press, 2015.

N. Nakashole, G. Weikum, and F. Suchanek. PATTY: A taxonomy of relational patterns with semantic types. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP), pages 1135–1145, Jeju Island, Korea, 2012. Association for Computational Linguistics.

C. Napoles, M. Gormley, and B. Van Durme. Annotated gigaword. In Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX), pages 95–100, Montréal, Canada, 2012. Association for Computational Linguistics.

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

R. Navigli and S. P. Ponzetto. BabelNet: Building a very large multilingual semantic network. In Proceedings of the 48th annual meeting of the Association for Computational Linguistics (ACL), pages 216–225, Uppsala, Sweden, 2010. Association for Computational Linguistics.

R. Navigli and S. P. Ponzetto. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. Artificial Intelligence, 193:217–250, 2012.

R. Navigli and P. Velardi. An analysis of ontology-based query expansion strategies. In Proceedings of the 14th European Conference on Machine Learning, Workshop on Adaptive Text Extraction and Mining, pages 42–49, Cavtat-Dubrovnik, Croatia, 2003.

S. Necsulescu, S. Mendes, D. Jurgens, N. Bel, and R. 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 (*SEM), pages 182–192, Denver, Colorado, 2015.

A. Neelakantan, J. Shankar, A. Passos, and A. McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1059–1069, Doha, Qatar, 2014. Association for Computational Linguistics.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. Advances in Neural Information Processing Systems 14, 2001.

H. T. Ng, B. Wang, and Y. S. Chan. Exploiting parallel texts for word sense disambiguation: An empirical study. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL), pages 455–462, Sapporo, Japan, 2003. Association for Computational Linguistics.

K. A. Nguyen, S. S. im Walde, and N. T. Vu. Distinguishing antonyms and synonyms in a pattern-based neural network. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL): Volume 1, Long Papers, volume 1, pages 76–85, Valencia, Spain, 2017. Association for Computational Linguistics.

Z.-Y. Niu, D.-H. Ji, and C.-L. Tan. I2r: Three systems for word sense discrimination, Chinese word sense disambiguation, and English word sense disambiguation. In Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007), pages 177–182, Prague, Czech Republic, 2007. Association for Computational Linguistics.

M. Palmer, H. T. Dang, and C. Fellbaum. Making fine-grained and coarse-grained sense distinctions, both manually and automatically. Natural Language Engineering, 13(2): 137–163, 2007.

A. Panchenko, E. Ruppert, S. Faralli, S. P. Ponzetto, and C. Biemann. Unsupervised does not mean uninterpretable: The case for word sense induction and disambiguation. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL): Volume 1, Long Papers, pages 86–98, Valencia, Spain, 2017. Association for Computational Linguistics.

B. Pang and L. Lee. Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval, 2(1-2):1–135, Jan. 2008.

P. Pantel and D. Lin. Discovering word senses from text. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 613–619. ACM, 2002.

P. Pantel and M. Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL), pages 113–120, Sydney, Australia, 2006. Association for Computational Linguistics.

P. Pantel and D. Ravichandran. Automatically Labeling Semantic Classes. In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL), pages 321–328, Boston, Massachusetts, 2004. Association for Computational Linguistics.

C. Paradis. Degree modifiers of adjectives in spoken british english. Lund Studies in English, 92, 1997.

R. J. Passonneau, A. Salleb-Aouissi, V. Bhardwaj, and N. Ide. Word sense annotation of polysemous words by multiple annotators. In Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10), Valletta, Malta, 2010. European Language Resources Association (ELRA).

E. Pavlick, J. Bos, M. Nissim, C. Beller, B. Van Durme, and C. Callison-Burch. Adding Semantics to Data-Driven Paraphrasing. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL/IJCNLP), pages 1512–1522, Beijing, China, 2015a. Association for Computational Linguistics.

E. Pavlick, P. Rastogi, J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL) (Volume 2: Short Papers), pages 425–430, Beijing, China, 2015b. Association for Computational Linguistics.

M. Paşca. Acquisition of categorized named entities for web search. In Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management (CIKM), pages 137–145, Washington, DC, 2004. ACM.

M. Paşca. Weakly-supervised discovery of named entities using web search queries. In Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management (CIKM), pages 683–690, Lisbon, Portugal, 2007. ACM.

K. Pearson. Principal components analysis. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 6(2):559, 1901.

T. Pedersen. UMND2: SenseClusters applied to the sense induction task of Senseval-4. In Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007), pages 394–397, Prague, Czech Republic, 2007. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

M. Pelevina, N. Arefiev, C. Biemann, and A. Panchenko. Making sense of word embeddings. In Proceedings of the 1st Workshop on Representation Learning for NLP (Rep4NLP), pages 174–183, Berlin, Germany, 2016. Association for Computational Linguistics.

J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar, 2014. Association for Computational Linguistics.

M. Peters, W. Ammar, C. Bhagavatula, and R. Power. Semi-supervised sequence tagging with bidirectional language models. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers), pages 1756–1765, Vancouver, Canada, 2017.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL) - Volume 1 (Long Papers), New Orleans, Louisiana, 2018. Association for Computational Linguistics.

T. Petrolito and F. Bond. A survey of WordNet annotated corpora. In Proceedings of the Seventh Global WordNet Conference, pages 236–245, Tartu, Estonia, 2014. University of Tartu Press.

A. Purandare and T. Pedersen. Word sense discrimination by clustering contexts in vector and similarity spaces. In Proceedings of the Eighth Conference oN Computational Natural Language Learning (CoNLL), pages 41–48, Boston, Massachusetts, 2004. Association for Computational Linguistics.

C. Quirk, C. Brockett, and W. Dolan. Monolingual machine translation for paraphrase generation. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 142–149, Barcelona, Spain, 2004. Association for Computational Linguistics.

K. Radinsky, E. Agichtein, E. Gabrilovich, and S. Markovitch. A word at a time: computing word relatedness using temporal semantic analysis. In Proceedings of the 20th international conference on World Wide Web (WWW), pages 337–346, Hyderabad, India, 2011. ACM.

S. Rajana, C. Callison-Burch, M. Apidianaki, and V. Shwartz. Learning antonyms with paraphrases and a morphology-aware neural network. In Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM), pages 12–21, Vancouver, Canada, 2017.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2383–2392, Austin, Texas, 2016. Association for Computational Linguistics.

J. Reisinger and R. J. Mooney. Multi-Prototype Vector-Space Models of Word Meaning. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), pages 109–117, Los Angeles, California, 2010. Association for Computational Linguistics.

P. Resnik and D. Yarowsky. Distinguishing Systems and Distinguishing Senses: New Evaluation Methods for Word Sense Disambiguation. Natural Language Engineering, 5(3): 113–133, 2000.

S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal, and Y. Liu. Statistical machine translation for query expansion in answer retrieval. In Proceedings of the 45th Annual Meeting of the Association For Computational Linguistics (ACL), pages 464–471, Prague, Czech Republic, 2007. Association for Computational Linguistics.

S. Rill, J. v. Scheidt, J. Drescher, O. Schütz, D. Reinel, and F. Wogenstein. A generic approach to generate opinion lists of phrases for opinion mining applications. In Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining (WISDOM), Beijing, China, 2012.

E. Riloff and J. Shepherd. A Corpus-Based Approach for Building Semantic Lexicons. In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 1997.

L. Rimell. Distributional Lexical Entailment by Topic Coherence. In Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL), pages 511–519, Gothenburg, Sweden, 2014. Association for Computational Linguistics.

A. Ritter, S. Soderland, and O. Etzioni. What Is This, Anyway: Automatic Hypernym Discovery. In Technical Report SS-09-07: Papers from the 2009 AAAI Spring Symposium, pages 88–93. AAAI Press, Menlo Park, California, 2009.

B. Roark and E. Charniak. Noun-phrase co-occurrence statistics for semiautomatic semantic lexicon construction. In Proceedings of the 17th International Conference on Computational Linguistics (COLING) - Volume 2, pages 1110–1116, Montréal, Canada, 1998. Association for Computational Linguistics.

S. Roller and K. Erk. PIC a Different Word: A Simple Model for Lexical Substitution in Context. In Proceedings of the 15th Annual Conference of the North American

Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pages 1121–1126, San Diego, California, 2016a. Association for Computational Linguistics.

S. Roller and K. 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 (EMNLP), pages 2163–2172, Austin, Texas, 2016b. Association for Computational Linguistics.

A. Rosenberg and J. Hirschberg. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In EMNLP-CoNLL, pages 410–420, Prague, Czech Republic, 2007. Association for Computational Linguistics.

S. Rothe and H. Schütze. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL) - Volume 1: Long Papers, pages 1793–1803, Beijing, China, 2015. Association for Computational Linguistics.

P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20:53 – 65, 1987.

H. Rubenstein and J. B. Goodenough. Contextual correlates of synonymy. Communications of the ACM, 8(10):627–633, 1965.

J. Ruppenhofer, M. Wiegand, and J. Brandes. Comparing methods for deriving intensity scores for adjectives. In Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL), pages 117–122, Gothenburg, Sweden, 2014. Association for Computational Linguistics.

J. Ruppenhofer, J. Brandes, P. Steiner, and M. Wiegand. Ordering adverbs by their scaling effect on adjective intensity. In Proceedings of the International Conference on Recent Advances in Natural Language Processing, pages 545–554, Hissar, Bulgaria, 2015. INCOMA Ltd. Shoumen, BULGARIA.

E. Santus, A. Lenci, Q. Lu, and S. S. 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 (EACL), pages 38–42, Gothenburg, Sweden, 2014. Association for Computational Linguistics.

E. Santus, F. Yung, A. Lenci, and C.-R. Huang. EVALution 1.0: an Evolving Semantic Dataset for Training and Evaluation of Distributional Semantic Models. In Proceedings of the 4th Workshop on Linked Data in Linguistics: Resources and Applications, pages 64–69, Beijing, China, 2015. Association for Computational Linguistics.

E. Santus, A. Lenci, T.-S. Chiu, Q. Lu, and C.-R. Huang. Nine features in a random forest to learn taxonomical semantic relations. Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC), pages 4557–4564, 2016.

H. Schütze. Dimensions of meaning. In Proceedings of the 1992 ACM/IEEE conference on Supercomputing, pages 787–796, Minneapolis, Minnesota, 1992. IEEE Computer Society Press.

H. Schütze. Automatic word sense discrimination. Computational linguistics, 24(1):97–123, 1998.

R. Sharma, M. Gupta, A. Agarwal, and P. Bhattacharyya. Adjective Intensity and Sentiment Analysis. In Proceedings of the 2015 Conference on Empirical Methods for Natural Language Processing (EMNLP), pages 2520–2526, Lisbon, Portugal, 2015. Association for Computational Linguistics.

V. Sheinman and T. Tokunaga. Adjscales: Visualizing differences between adjectives for language learners. IEICE TRANSACTIONS on Information and Systems, 92(8):1542–1550, 2009.

V. Sheinman, C. Fellbaum, I. Julien, P. Schulam, and T. Tokunaga. Large, huge or gigantic? Identifying and encoding intensity relations among adjectives in WordNet. Language Resources and Evaluation, 47(3):797–816, 2013.

C. P. Shivade, M.-C. de Marneffe, E. Fosler-Lussier, and A. M. Lai. Corpus-based discovery of semantic intensity scales. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), pages 483–493, Denver, Colorado, 2015. The Association for Computational Linguistics.

V. Shwartz and I. Dagan. Adding context to semantic data-driven paraphrasing. In Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics, pages 108–113, Berlin, Germany, 2016a. Association for Computational Linguistics.

V. Shwartz and I. Dagan. Path-based vs. Distributional Information in Recognizing Lexical Semantic Relations. Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogAlex-V), pages 24–29, 2016b.

V. Shwartz, E. Santus, and D. Schlechtweg. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL), pages 65–75, 2017.

R. Snow, D. Jurafsky, and A. Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In Advances in Neural Information Processing Systems 18, pages 1297–1304, 2005.

R. Snow, D. Jurafsky, and A. 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 (ACL), pages 801–808, Sydney, Australia, 2006. Association for Computational Linguistics.

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

S. Šuster, I. Titov, and G. van Noord. Bilingual learning of multi-sense embeddings with discrete autoencoders. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), pages 1346–1356, San Diego, California, 2016. Association for Computational Linguistics.

M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. Lexicon-based methods for sentiment analysis. Computational linguistics, 37(2):267–307, 2011.

B. Thorsten and A. Franz. Web 1t 5-gram version 1 ldc2006t13. DVD. Philadelphia: Linguistic Data Consortium, 2006.

D. H. Tuggy. Ambiguity, Polysemy and Vagueness. Cognitive linguistics, 4(2):273–290, 1993.

P. D. Turney. Domain and function: A dual-space model of semantic relations and compositions. Journal of Artificial Intelligence Research, 44:533–585, 2012.

P. D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. Journal of Artificial Intelligence Research, 37:141–188, 2010.

S. Upadhyay, K.-W. Chang, M. Taddy, A. Kalai, and J. Zou. Beyond bilingual: Multi-sense word embeddings using multilingual context. In Proceedings of the 2nd Workshop on Representation Learning for NLP (RepL4NLP), pages 101–110, Vancouver, Canada, 2017. Association for Computational Linguistics.

D. Ustalov, A. Panchenko, and C. Biemann. Watset: Automatic Induction of Synsets from a Graph of Synonyms. In Proceedings of the 55th Meeting of the Association for Computational Linguistics (ACL), pages 1579–1590, Vancouver, Canada, 2017. Association for Computational Linguistics.

L. Van der Plas and J. Tiedemann. Finding synonyms using automatic word alignment and measures of distributional similarity. In Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 866–873, Sydney, Australia, 2006. Association for Computational Linguistics.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, . Kaiser, and I. Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems 30, pages 5998–6008, Long Beach, California, 2017. Curran Associates, Inc.

I. Vendrov, R. Kiros, S. Fidler, and R. Urtasun. Order-embeddings of images and language. In Proceedings of the 4th International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2016.

J. Véronis. Hyperlex: lexical cartography for information retrieval. Computer Speech & Language, 18(3):223–252, 2004.

P. Vossen. Eurowordnet: a multilingual database of autonomous and language-specific wordnets connected via an inter-lingualindex. International Journal of Lexicography, 17 (2):161–173, 2004.

Y. Vyas and M. Carpuat. Detecting Asymmetric Semantic Relations in Context: A Case-Study on Hypernymy Detection. In Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM), pages 33–43, Vancouver, Canada, 2017. Association for Computational Linguistics.

W. Weaver. Translation. In W. N. Locke and A. D. Boothe, editors, Machine Translation of Languages, pages 15–23. MIT Press, Cambridge, MA, 1955. Reprinted from a memorandum written by Weaver in 1949.

J. Weeds and D. Weir. A general framework for distributional similarity. In Proceedings of the 2003 conference on Empirical Methods in Natural Language Processing (EMNLP), pages 81–88, Sapporo, Japan, 2003. Association for Computational Linguistics.

J. Weeds, D. Clarke, J. Reffin, D. Weir, and B. Keller. Learning to distinguish hypernyms and co-hyponyms. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 2249–2259, Dublin, Ireland, 2014.

R. Weischedel, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue, A. Taylor, J. Kaufman, M. Franchini, et al. Ontonotes release 5.0 ldc2013t19. Linguistic Data Consortium, Philadelphia, PA, 2013.

B. Wilkinson. Identifying and Ordering Scalar Adjectives Using Lexical Substitution. PhD Thesis, University of Maryland, Baltimore County, 2017.

B. Wilkinson and T. Oates. A Gold Standard for Scalar Adjectives. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC), Portorož, Slovenia, 2016. European Language Resources Association (ELRA).

W. Xiong, T. Hoang, and W. Y. Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 564–573, Copenhagen, Denmark, 2017. Association for Computational Linguistics.

D. Yang and D. M. Powers. Measuring semantic similarity in the taxonomy of WordNet. In Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38, pages 315–322. Australian Computer Society, Inc., 2005.

H. Yang and J. Callan. A metric-based framework for automatic taxonomy induction. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 271–279, Suntec, Singapore, 2009. Association for Computational Linguistics.

X. Yao and B. Van Durme. Nonparametric bayesian word sense induction. In Proceedings of TextGraphs-6: Graph-based Methods for Natural Language Processing, pages 10–14, Portland, Oregon, 2011. Association for Computational Linguistics.

X. Yao, B. Van Durme, and C. Callison-Burch. Expectations of Word Sense in Parallel Corpora. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), pages 621–625, Montréal, Canada, 2012. Association for Computational Linguistics.

M. A. Yatbaz, E. Sert, and D. Yuret. Learning syntactic categories using paradigmatic representations of word context. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 940–951, Jeju Island, Korea, 2012. Association for Computational Linguistics.

K. Yu, S. Yu, and V. Tresp. Soft clustering on graphs. In Advances in Neural Information Processing Systems 18, pages 1553–1560. MIT Press, 2005.

L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In Advances in Neural Information Processing Systems 17, pages 1601–1608. MIT Press, 2004.

Z. Zhong and H. T. Ng. It makes sense: A wide-coverage word sense disambiguation system for free text. Proceedings of the ACL 2010 System Demonstrations, pages 78–83, 2010.

D. Zhou, T. Hofmann, and B. Schölkopf. Semi-supervised learning on directed graphs. In Advances in Neural Information Processing Systems 17, pages 1633–1640. MIT Press, 2004.

J. Zhou and W. Xu. End-to-end learning of semantic role labeling using recurrent neural networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL) (Volume 1: Long Papers), pages 1127–1137, Beijing, China, 2015. Association for Computational Linguistics.

R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pages 45–50, Valletta, Malta, 2010. European Language Resources Association (ELRA).