

Identifying Lexical Relationships and Entailments with Distributional Semantics

Stephen Roller
The University of Texas at Austin
roller@cs.utexas.edu

Doctoral Dissertation Proposal

September 7, 2016

Abstract

As the field of Natural Language Processing has developed, research has progressed on ambitious semantic tasks like Recognizing Textual Entailment (RTE). Systems that approach these tasks may perform sophisticated inference between sentences, but often depend heavily on lexical resources like WordNet to provide critical information about relationships and entailments between lexical items. However, lexical resources are expensive to create and maintain, and are never fully comprehensive.

Distributional Semantics has long provided a method to automatically induce meaning representations for lexical items from large corpora with little or no annotation efforts. The resulting representations are excellent as proxies of semantic similarity: words will have similar representations if their semantic meanings are similar. Yet, knowing two words are similar does not tell us their relationship or whether one entails the other.

We present several models for identifying specific relationships and entailments from distributional representations of lexical semantics. Broadly, this work falls into two distinct but related areas: the first predicts specific ontology relations and entailment decisions between lexical items devoid of context; and the second predicts specific lexical paraphrases in complete sentences. We provide insight and analysis of how and why our models are able to generalize to novel lexical items and improve upon prior work.

We propose several short- and long-term extensions to our work. In the short term, we propose applying one of our hypernymy-detection models to other relationships and evaluating our more recent work in an end-to-end RTE system. In the long-term, we propose adding consistency constraints to our lexical relationship prediction, better integration of context into our lexical paraphrase model, and new distributional models for improving word representations.

Contents

1	Introduction	3
2	Background and Related Work	5
2.1	Recognizing Textual Entailment	5
2.2	Distributional Semantics	5
2.3	Lexical Entailment and Relationship Detection	8
2.4	Lexical Substitution	9
3	Completed work	10
3.1	Asym Model for Lexical Entailment (Roller et al., 2014)	10
3.2	Subsystem in complete RTE system (Beltagy et al., 2016)	13
3.3	H-Features for Hypernymy Classification (Roller and Erk, 2016b)	16
3.3.1	Concerning Lexical Memorization	16
3.3.2	The H-Feature Detector Model	18
3.4	Lexical Substitution (Roller and Erk, 2016a)	21
4	Proposed Work	23
4.1	Short Term Proposals	24
4.1.1	Lexical Entailment	24
4.1.2	Lexical Substitution in RTE	25
4.2	Long Term Proposals	26
4.2.1	Ontology Constraints in Hypernymy Prediction	26
4.2.2	Sophisticated Contexts for Lexical Substitution	27
4.2.3	Generalized Distributional Spaces for Entailment and Substitution	30
5	Conclusion	32

1 Introduction

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

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

Text (Antecedent): The bright girl reads a book.

Hypothesis (Consequent): A smart child looks at pages of text.

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

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

Distributional Semantics offers one potential solution to these issues of lexical coverage. Distributional Semantics takes inspiration from the famous quote: “You shall know a word by the company it keeps” (Firth, 1957). In Distributional Semantics, representations of word meaning are automatically induced by counting or modeling the *contexts* in which a word appears. Distributional Semantics is often called Vector Space Models (VSMs) of language, because words are represented as vectors a high-dimensional vector space. Words with similar semantics will have

similar vectors in this space. Since VSMs do not require annotated corpora, they are used and studied as an alternative or predictor of particular lexical resources (Baroni et al., 2012; Erk and Padó, 2008; Turney and Pantel, 2010).

In our work, we consider how VSMs can be leveraged to predict some of the lexical inferences necessary in RTE. Namely, we present techniques and models for predicting specific lexical relationships, entailments, and substitutions using Distributional Semantics. In Lexical Relationship Detection, we must predict whether two words exhibit specific relationships, like hypernymy (is-a relationships) or meronymy (has-a relationships). This is sometimes supplanted by Lexical Entailment Detection, where we must predict a coarser entailment prediction. We present two original models which can learn to predict hypernymy or general entailment relations (Roller et al., 2014; Beltagy et al., 2016; Roller and Erk, 2016b), and evaluate their performance on different datasets. In these works, we also make significant contributions in experimental setups which prevent issue with lexical memorization (Roller et al., 2014), and insight into how these models work (Roller and Erk, 2016b). We also present an original model for Lexical Substitution, where one must predict a context-specific synonym for a given target word in a sentential context (Roller and Erk, 2016a).

Finally, we propose several short- and long-term extensions to our completed work. In the short-term, we focus mainly on how our more recent work may be expanded to lexical relationships other than hypernymy, and how our recent publications may contribute as a component in an end-to-end RTE system, which would cement the connection between our work and Textual Entailment. In the long-term, we propose three broad directions forward: (1) providing better internal consistency in the predictions made by our system; (2) integration of larger contexts into our Lexical Substitution model; and (3) improved models of distributional semantics which more efficiently use syntactic information.

2 Background and Related Work

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

2.1 Recognizing Textual Entailment

In the Introduction, we introduced the Recognizing Textual Entailment (RTE) task as a long standing, challenging semantic problem in the field of Natural Language Processing. One of the first benchmark papers describes RTE as “recognizing, given two text fragments, whether the meaning of one text can be inferred (entailed) from the other” (Dagan et al., 2006). Since this original definition, many other datasets (Giampiccolo et al., 2007; Bentivogli et al., 2009; Marelli et al., 2014) and countless approaches have been described (c.f. Dagan et al. (2013) for a thorough survey). Although RTE is not usually considered an end-user application by itself, successful RTE systems could influence many downstream tasks like Information Extraction or Question Answering and become useful in information-heavy industries like defense, journalism, and science.

RTE is a very difficult task, at least partially due to its generality. Entailment reasoning may require: broad common sense knowledge or highly specialized domain knowledge; sophisticated logical inferences about quantifiers and implication; or more graded, fuzzier reasoning about related words. In our own work, we focus predominantly on some of the issues of *lexical semantics* necessary for reasoning in RTE systems. These include issues like lexical relationship detection, where one must classify *how* two words are (or are not) related, and lexical paraphrasing, where one must suggest alternative words which have the same meaning.

Presently, RTE systems often employ a wide collection of lexical resources in order to capture some of these issues in lexical semantics (MacCartney and Manning, 2008; Bjerva et al., 2014; Beltagy et al., 2016). These rich lexical resources, including WordNet (Miller, 1995) and PPDB (Ganitkevitch et al., 2013), provide an excellent source of common sense knowledge and word relationships which can be used as a background knowledge-base during logical reasoning. In our work, we consider whether it is possible to distill information about lexical relationships automatically. We turn now to Distributional Semantics and Vector Space Models, which provide an automatic induction of word meaning using only large, unannotated corpora.

2.2 Distributional Semantics

Distributional Semantics is a powerful tool for automatically inducing semantic representations for lexical items (Turney and Pantel, 2010; Erk, 2012). The core notion is that of the *Distributional Hypothesis*, that if two words appear in similar *contexts*, they can be assumed to have similar meaning. This idea has a long history in the linguistic and philosophical literature that can be

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

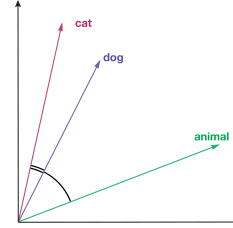


Figure 1: (a) Example contexts of the word *dog*, and (b) a cartoon drawing of word vectors.

traced back over 60 years (Wittgenstein, 1953; Harris, 1954; Firth, 1957). In its modern form, Distributional Semantics involves finding *vector space representations* of words which are constructed by counting or modeling the contexts in which a particular word appears. According to the Distributional Hypothesis, words with similar *vectors* can be assumed to have similar *meanings* (Turney and Pantel, 2010). For this reason, they are often referred to as Vector Space Models (VSMs) of language. Variations on this idea have also become immensely popular in the neural networks community, with algorithms like Skip-gram Negative Sampling (SGNS) (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), and have often replaced traditional count-based VSMs in the NLP community (Baroni et al., 2014b).

In its simplest form, vectors are induced by defining a vector space where each dimension in the space corresponds to a particular context word. A large, unannotated corpus of text is then processed, finding instances of a target word, like *dog*, and incrementing a count for each of the target’s *co-occurrences*, or words appearing around the target word *dog*, as in Figure 1. With a large enough corpus, coherent statistical patterns begin to form. For example, the word *furry* is likely to be used to describe both *cat* and *dog*, which is then reflected in the vector counts (Lund and Burgess, 1996). After constructing vector representations for the words *cat* and *dog*, we can then compare these vectors using various geometric distance metrics, most prominently *cosine similarity*:

$$\text{cosine}(u, v) = \frac{\sum_i u_i v_i}{\sqrt{\sum_i u_i^2 \sum_i v_i^2}} \quad (1)$$

Here, i iterates over all the different context dimensions, like *furry* or *kennel*, and cosine similarity is defined over the range $[-1, 1]$. Words with similar vectors will have a smaller angle between them, and therefore a higher cosine similarity (i.e. close to 1).

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

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

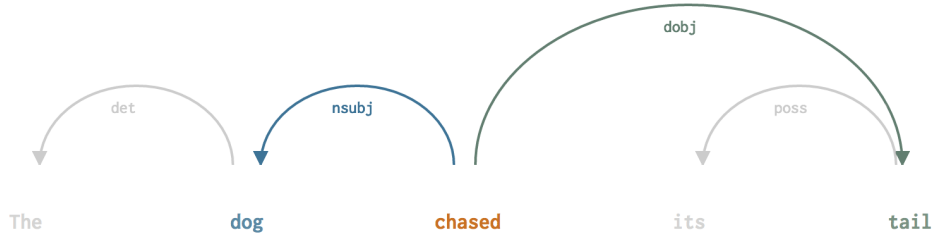


Figure 2: Example of a dependency parse for “The dog chased its tail.” In a syntactic distributional space, contexts are defined as adjacent nodes with their labeled edges.

Pointwise Mutual Information (PMI) measures roughly how many times more likely two items co-occur more often than chance, while Positive PMI additionally ignores co-occurrences that occur less often than chance. Other transformations, like conditional probability (Hofmann, 1999; Blei et al., 2003) and Softplus (Pennington et al., 2014), are also sometimes seen in the literature, and emphasize different aspects of lexical similarity.

Syntactic Contexts Defining contexts is another important aspect of Distributional Semantics. In the example of Figure 1, we showed that context can be defined as three words to the left and right of the target word, but there are alternatives. For example, using very large windows of co-occurrence (or even entire documents) results in emphasizing more *topical* similarity, e.g. doctor and hospital, while smaller windows emphasize more *functional* similarity, e.g. doctor and surgeon (Padó and Lapata, 2007; Erk and Padó, 2008; Levy and Goldberg, 2014a).

Context can be also defined as *syntactic neighbors* extracted from a dependency parse. For example, in Figure 2, the contexts for the word *chased* would be *nsubj+dog* and *dobj+tail*. Distributional Spaces defined in this manner tend to emphasize the *selectional preferences* of words, or the tendency of words to have particular arguments in their syntactic relations. (Padó and Lapata, 2007; Erk and Padó, 2008; Baroni and Lenci, 2010; Levy and Goldberg, 2014a). For example, the subject of *barks* is likely to be *dog*, while the subject of *purrs* is likely to be *cat*.

Dimensionality Reduction Dimensionality Reduction is another important aspect of Distributional Semantics. As described earlier, distributional vector spaces are very high-dimensional: bag-of-words spaces have many thousands of dimensions (Turney and Pantel, 2010; Mikolov et al., 2013; Pennington et al., 2014), while syntactic spaces usually have a millions (Baroni and Lenci, 2010). Efficiently dealing with these large, extremely sparse vectors can be troublesome, so we often opt to use some form of *dimensionality reduction*, like Singular Value Decomposition (SVD) (Deerwester et al., 1990; Landauer and Dumais, 1997) or Nonnegative Matrix Factorization (NNMF) (Lee and Seung, 2001). In dimensionality reduction, the co-occurrence matrix M is typically assumed to be factorizable into two lower-rank matrices,

$$M = VC^T \tag{2}$$

where V is some lower dimension representation of word vectors, and C is the corresponding lower dimension representation of the context items. These projections of words and contexts into

the same latent space traces back to the earliest days of distributional semantics (Deerwester et al., 1990), and is critical to many of the contributions of our completed work. Interestingly, the most popular algorithms for computing word embeddings, like Skip-Gram Negative Sampling (SGNS) (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) can be viewed as form of dimensionality reduction (Levy and Goldberg, 2014b; Levy et al., 2015a).

2.3 Lexical Entailment and Relationship Detection

To paraphrase Shnarch (2008), Lexical Entailment may be broadly defined as any number of semantic relations between two lexical items where the meaning of one is implied by the meaning of another. This includes many classical lexical relations, like hypernymy (*a girl is a child; a dog is an animal*), and meronymy (*a girl has eyes; a dog has a tail*), but it can also include a wide variety of other inferences which are difficult to categorize, like *to snore implies to sleep*.

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

One important, early development in this task was Hearst patterns (Hearst, 1992), which are specific textual patterns highly indicative of particular relationships. Common Hearst patterns include exemplar phrases like “X such as Y,” “X including Y,” which are both highly indicative of hypernymy. Possessive phrases, like “X’s Y”, can be indicative of meronymy. Later, Snow et al. (2004) extended this Hearst pattern approach to use syntactic patterns. By using syntactic parses, some longer distance patterns are more easily captured, like “X such as Y and Z,” which implies “X such as Z.”

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

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

the Distributional Inclusion Hypothesis in one form or another (Clarke, 2009), or a hybrid of DIH and cosine similarity (Kotlerman et al., 2010; Lenci and Benotto, 2012).

As it became obvious that unsupervised measures did not work as well as hoped, the community began working on entailment detection as a supervised task. Baroni et al. (2012) proposed, as a preliminary baseline of a novel dataset, training a simple baseline classifier to predict whether word pairs were either hypernyms or non-hypernyms. Although they reported strong performance, others later realized their model struggled with issues of *lexical memorization*, or a special kind of overfitting (Roller et al., 2014; Weeds et al., 2014; Levy et al., 2015b). As such, more recent works have emphasized their performance when *individual words are held out entirely*, so that the same word can never appear in both training and testing sets (Roller et al., 2014; Kruszewski et al., 2015; Levy et al., 2015b; Shwartz et al., 2016; Roller and Erk, 2016a). We discuss more about this issue of Lexical Memorization in Section 3.3.1.

2.4 Lexical Substitution

In our discussion of lexical relationships, we assumed that words are mononymous, or that they have only one meaning. However, words like “bright” have multiple meanings, which change depending on context, as in “bright girl” and “bright coat.” This is called *polysemy*, and is a major issue in Natural Language Understanding, since it adds another layer of ambiguity.

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

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

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

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

Distributional Semantics offers a tempting solution to this problem of Lexical Substitution, given its ability to measure the *graded* levels of similarity between words (Erk and Padó, 2008).

Interestingly, although there have been both supervised (Biemann, 2012; Szarvas et al., 2013) and unsupervised attempts (Erk and Padó, 2008; Dinu and Lapata, 2010; Thater et al., 2010; Van de Cruys et al., 2011; Kremer et al., 2014; Melamud et al., 2015a; Melamud et al., 2015b; Kawakami and Dyer, 2016; Roller and Erk, 2016a) using distributional semantics, presently unsupervised measures hold a lead (Melamud et al., 2015a; Melamud et al., 2016).

3 Completed work

In this section, we discuss our publications related to this proposal, and emphasize our major contributions to the field. In short, we discuss two models for hypernymy detection, and compare and contrast them other work in the literature. We also discuss how one of these models can contribute as one component in an end-to-end RTE system. Finally, we discuss our model for the Lexical Substitution task, and why it improves upon prior work.

3.1 Asym Model for Lexical Entailment (Roller et al., 2014)

Most baseline similarity measures in distributional semantics, like cosine, have the unfortunate property that they are symmetric: $\text{cosine}(a, b) = \text{cosine}(b, a)$. While this is often desirable, it is a fatal flaw in any application involving lexical entailment: although *girl* implies *child*, but the opposite does not hold. As such, attempts to predict lexical entailment using solely symmetric measures will always fall short.

This has been recognized widely in the literature for some time, and numerous asymmetric, unsupervised similarity measures have been proposed (Weeds and Weir, 2003; Zhitomirsky-Geffet and Dagan, 2005; Clarke, 2009; Kotlerman et al., 2010; Santus, 2013), mostly inspired by the Distributional Inclusion Hypothesis (DIH), which states that the contexts of a hypernym should be a superset of its hyponyms'. However, their performance tend to be lackluster (Clarke, 2009) or brittle (Kotlerman et al., 2010). This raises the question: are the measures just overly sensitive to noise in distributional vectors, or is the Distributional Inclusion Hypothesis fundamentally flawed? If the unsupervised measures are simply too sensitive to noise, perhaps using supervised techniques can improve performance. To this end, we propose Asym, a simple supervised model that is inherent asymmetric and interpretable under the DIH (Roller et al., 2014).

At its core, the model is inspired by the famous result of Mikolov et al. (2013), who observed that vector subtraction can be used to perform some kinds of analogical reasoning in some kinds of distributional spaces: e.g., $\text{king} - \text{man} + \text{woman} \approx \text{queen}$. Interestingly, this vector subtraction approach reasonably models many grammatical relationships (singular/plural, verb conjugations) and some limited semantic relationships (gender, capital/country). Asym exploits this behavior for the task of hypernymy and lexical relationship prediction.

The Asym model is a simple model which uses the *vector difference* between the hypothesized hypernym-hyponym pair as input features to an off-the-shelf classifier. For example, given a (unit normalized) distributional vector for *animal* and a vector for *cat*, we use the vector $\text{animal} - \text{cat}$ as a positive example, while the vectors for $\text{cat} - \text{animal}$ and $\text{animal} - \text{sofa}$ are inputted as negative examples. Additionally, we also give the *element-wise squared difference vector* as features to the

classifier. Formally, for a given (hypernym, hyponym) pair of words, (H, w) , we compute the final feature space defined as:

$$\begin{aligned} A_i(H, w) &= H_i - w_i \\ B_i(H, w) &= (H_i - w_i)^2 \\ \text{features}(H, w) &= \langle A; B \rangle, \end{aligned}$$

where $\langle A; B \rangle$ is the vector *concatenation*. This computation is performed for all examples in our dataset, and then the features (H, w) vector and the classification label are used to train a Logistic Regression classifier.

One significant advantage of this model over other works is its direct connection to the Distributional Inclusion Hypothesis: since our model uses the vector difference as input, it naturally measures whether H_i is greater than w_i , effectively acting as a strict-subset measurement. The difference-squared part of the input features measures whether they have a large absolute difference, effectively capturing “equal” part of the “less than or equal” relation. As such, one interpretation of the model is a kind of *Selective Distributional Inclusion Hypothesis*, which presupposes that the DIH holds, but only in particular, relevant dimensions.

To evaluate our model, we train and measure accuracy of the Asym model on two datasets in a variation of leave-one-out cross validation (LOOCV) and measuring absolute accuracy. In this variation of LOOCV, we select one word from the vocabulary in the datasets, and consider *all pairs* with that word to be test pairs. The remainder of word pairs, which do not contain the held out word, are treated as training pairs. This prevent classifiers from simply memorizing that words like *animal* are more likely to be hypernyms. This experimental setup is one of our core contributions to the literature, as we were the first to recognize this problem and propose an experimental setup which avoids it (Roller et al., 2014). We revisit this issue in more detail in Section 3.3.1.

Since different types of distributional spaces exhibit different properties (Padó and Lapata, 2007), we evaluate our model on two distributional spaces which use a simple Bag-of-Words context. The *Window-2 BoW* space counts content words two words to the left and right of targets as contexts, while the *Sentence BoW* space counts all content words within complete sentence boundaries. Both spaces are reduced to 300 dimensions using the Singular Value Decomposition (Landauer and Dumais, 1997).

We evaluate our model on two datasets. The first, **LEDS** (Baroni et al., 2012), contains 1385 hyponym-hypernym pairs as positive examples and 1385 negative pairs which were generated by randomly shuffling the positive examples. As such the model only contains hypernymy and random relations, and we train a binary classifier. The second dataset is **BLESS** (Baroni and Lenci, 2011), which contains annotations of word relations for 200 unambiguous, concrete nouns from 17 broad categories. Each noun is annotated with its co-hyponyms, meronyms, hypernym and some random words. Since there are four relations, we train four one-vs-all classifiers, and predict the relation with the highest score; in this way, the model actually learns to detect three different lexical relations, though this was not our primary research interest at the time. We will reconsider this in our proposed work.

We compare our model with two baselines: the first is a degenerate baseline, which guesses false for the (balanced) LEDS dataset, and always the most common label (*no-relation*) for BLESS.

Classifier	Space	LEDS	BLESS
Always guess false/no relation	-	.50	.46
(Baroni et al., 2012)	Window 2	.81	.76
Asym (Roller et al., 2014)	Window 2	.85	.84
(Baroni et al., 2012)	Sentence	.78	.73
Asym (Roller et al., 2014)	Sentence	.82	.80

Table 1: Accuracy of Baroni et al. (2012) and Roller et al. (2014) on BLESS and LEDS using different spaces for feature generation. Performance is measured as the average accuracy across all folds of the leave-one-out cross validation experiment.

We also compare to the model proposed in Baroni et al. (2012), which uses the *concatenation* of the H and w vectors and trains an off-the-shelf polynomial Support Vector Machine (Cortes and Vapnik, 1995).

Table 1 shows the results for our initial experiment. First we notice that both models strongly outperform the degenerate baseline, indicating there is some successful learning in the models. We also see that the Window 2 space performs better than the Sentence space in all four comparisons, indicating it is likely the task depends more heavily *functional* properties of words than *topical* properties of words.

Finally, we see that the Asym model outperforms the model proposed by Baroni et al. (2012) in all four comparisons, indicating our architecture has better lexical generalization. Interestingly, we found that dropping the square-difference terms severely hurt the performance of our model, emphasizing these features immense importance. We will discuss more of why these features are so important in Section 3.3.1. Incidentally, at the same time that Roller et al. (2014) was published, Weeds et al. (2014) and Fu et al. (2014) also proposed supervised hypernymy models based on vector difference, but neither of these employ the critical square-difference terms, or adequately address the issue of lexical memorization.

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

We measure every word pair’s similarity using three similarity measures: cosine, Clarke, and invCL. Cosine similarity acts as our scientific control, and should *not* change substantially between the original and selective spaces, while the others, which are based on Distributional Inclusion, should. The second similarity measure, *Clarke*, measures roughly what percentage of the

	Original Space				Selective Space			
Measure	Co-hyp	Hyper	Mero	Random	Co-hyp	Hyper	Mero	Random
cosine	.68	.20	.27	.27	.69	.20	.24	.28
Clarke	.66	.19	.28	.28	.55	.39	.24	.29
invCL	.60	.18	.31	.28	.42	.58	.24	.29

Table 2: Mean Average Precision for the unsupervised measures before after selecting the top dimensions from the Asym model.

hyponyms’ mass is contained within the hypernym (Clarke, 2009):

$$\text{Clarke}(H, w) = \frac{\sum_i \min(H_i, w_i)}{\sum_i H_i};$$

The final similarity measure, *invCL*, extends Clarke to additionally measure what percentage of the hypernym’s mass is *not* contained within the hyponym (Lenci and Benotto, 2012), extending Clarke to roughly measure *strict* containment:

$$\text{invCL}(H, w) = \sqrt{\text{Clarke}(H, w)(1 - \text{Clarke}(w, H))}.$$

We compute all three similarity measures across all the word pairs in BLESS, and computed Mean Average Precision (MAP) across all pairs for each measure and distributional space. Ideally, we should see that, compared to the original space, the selective space has higher Clarke and *invCL* values for hypernyms, and lower Clarke and *invCL* values for the other relations. Table 2 shows the results of this experiment.

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

3.2 Subsystem in complete RTE system (Beltagy et al., 2016)

Beyond showing that Asym is better able to improve performance on lexical relationship datasets, we should also show that it can improve performance in an end-to-end Recognizing Textual Entailment (RTE) system. Specifically, we compare Asym’s performance to a variety of lexical entailment classifiers which use a variety of hand-engineered features and word lexicons. These predictions made by lexical entailment classifiers are used as to generate *logical rules*; an inference engine based on Markov Logic Networks (MLNs) is then used to form predictions about complete *sentential* entailment (Beltagy et al., 2016).

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

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

Table 3: Example entailing and contradicting sentences from the SICK dataset.

To this end, we employ the Sentences Involving Compositional Knowledge (SICK) dataset, which contains nearly 10k sentence pairs, evenly split between training and test sets (Marelli et al., 2014). Sentence pairs were extracted randomly image caption datasets, and then simplified and extended to cover semantic issues like negation and quantifiers, and then manually annotated as *entailing* (the antecedent implies the consequent), *contradicting* (the antecedent implies the opposite of the consequent), or *neutral* (neither of the above). Two examples from the dataset are shown in Table 3.

To train the lexical entailment classifier, we extract *lexical pairs* from the SICK dataset using a variation on Robinson Resolution (Robinson, 1965). The full details are outside the scope of this document, but briefly, we employ an off-the-shelf semantic parser called Boxer, which translates sentences into First Order Logical formulas (Bos, 2008). We then use theorem proving techniques in order to extract lexical rules which *must* be true in the dataset. For example, given that “The girl talks” entails “A girl speaks,” we can eliminate *girl* and automatically conclude that *talks* lexically entails *speaks*.

Crucially, by knowing entailment decisions about the entire sentences, and by performing unification over their logical structures, we can use theorem proving to automatically label certain *atomic rules* as entailing, contracting, or neutral. These atomic rules, like *talks entails speaks*, can be interpreted as *lexical entailment pairs* for use in a lexical entailment classifier. Most, but not all, pairs can be labeled automatically, and those that cannot are manually annotated by two of the authors (Beltagy et al., 2016). The final result is a novel dataset of lexical entailment pairs, which we call RRR (Robinson Resolution Rules), which we use to train and compare lexical entailment classifiers.

We compare the performance of several existing lexical entailment classifiers which employ a variety of hand-engineered features and lexicons like WordNet. Most of the hand-engineered features come from Lai and Hockenmaier (2014), and include things like Wordform features (e.g., do the words have the same lemma or Part-of-Speech?); WordNet features (e.g., are the words hypernyms or co-hyponyms in WordNet?); and Distributional Similarity (e.g., cosine distance between two words in a distributional space).

We also present a novel extension of the real-valued distributional similarity features, by *binning* cosines into ranges (e.g. 0.0–0.1, . . . , 0.9–1.0) to transform them into binary-valued features, after observing that mid-similar terms (those with a cosine of $\sim .80$, like *cat* and *animal*) were more likely entailments than those with high similarity (cosine of $\sim .95$, like *cat* and *dog*). We found this binning technique significantly improved the contribution distributional similarity in feature-engineered lexical entailment classifiers.

Feature set	Intrinsic	RTE Test
Always guess neutral	56.6	69.3
Gold standard annotations	100.0	94.6
Wordform only	57.4	70.4
Dist. Sim. only	68.8	76.7
WordNet only	79.1	84.2
Asym only	76.8	79.2
Asym + Concat	81.4	82.6
All features	84.6	83.8

Table 4: Accuracies for various Lexical Entailment classifiers on the RRR (Intrinsic) and SICK (RTE) datasets.

Finally, we also compare to the Asym classifier trained on our new RRR dataset, and on a variation of Asym which concatenates the Asym features with the antecedent (LHS) and consequent (RHS) vectors, which we call Asym + Concat, since it uses both the Asym features and the Concat (LHS+RHS) features. For both the distributional similarity features, and the Asym models, we use two distributional spaces: one which uses a BoW window of two words, and one based on syntactic contexts.

We evaluate all of the lexical entailment classifiers listed above on two variations of the task: Intrinsic accuracy, and RTE accuracy. In the Intrinsic setup, the lexical entailment classifiers are evaluated on their performance on the RRR dataset in a standard 10-fold cross-validation setup. In the RTE setup, the classifiers are trained on items extracted from the training portion of the RTE dataset, and then used as the sole source of lexical knowledge in a complete RTE pipeline Beltagy et al. (2013; Beltagy et al. (2016) to predict the test portion of the RTE dataset. This ensures we are actually testing whether Asym is able to contribute in a rich RTE pipeline.

Table 4 shows performance of the various lexical entailment classifiers on each of our evaluation settings. We also report a degenerate baseline (always guess Neutral), and an upper baseline, which always gives gold lexical entailment decisions. Note that this upper baseline on the RTE task is *not* 100%, due to a mixture of issues like parsing errors, inference errors, and other issues in the RTE inference engine.

In general, we find that Asym performs relatively well, even compared some of the hand-engineered features proposed by Lai and Hockenmaier (2014), indicating that Asym is flexible and able to learn beyond just the LEDS and BLESS datasets. Unsurprisingly, the classifiers which use only Wordform features or only distributional similarity both perform much worse than the information-rich WordNet features, or the more sophisticated Asym classifier. We also notice that the classifier which combines Asym with the Concat vectors performs substantially better than Asym does by itself, indicating a need for additional study, which we will address in Section 3.3.

Unsurprisingly, we find that the WordNet classifier does a bit better than the others, which is expected given that WordNet is an information-rich resource, and contains gold annotations about word relationships, rather than distributional information. Finally, we observe a classifier which combines all of these features together does the best on the Intrinsic accuracy, but not as strongly

as WordNet on the end-to-end task; some of this can be attributed to a handful of systematic differences between the training and test sets of SICK.

We also perform a qualitative analysis to compare the Distributional Similarity classifier with the Asym classifier. We manually inspect and compare the predictions and errors made by each, and find that Asym does substantially better at distinguishing hypernymy from co-hyponymy. This is what we had hoped to find, given the findings in Section 3.1, and that cosine is known to heavily favor co-hyponymy (Baroni and Lenci, 2011). However, we also find that cosine features are better at discovering synonymy, and that Asymmetric frequently mistakes antonymy as an entailing. Additional qualitative analyses comparing models are available in Beltagy et al. (2016).

3.3 H-Features for Hypernymy Classification (Roller and Erk, 2016b)

In the previous sections, we saw that the Asym classifier is able to reasonably learn to classify word pairs as hypernymy and non-hypernymy, and that is able to contribute in an end-to-end RTE system. However, we also saw in our RTE experiments that Asym can be improved upon by simply concatenating the Asym difference vectors with vectors for the LHS and the RHS (which we call Concat). In this section, we discuss some of the strengths and weaknesses of the Concat model, and how these relate to the Asym model. We then propose a novel classification model which combines and extends the strengths of all these models using an iterative procedure similar to Principal Component Analysis (PCA).

3.3.1 Concerning Lexical Memorization

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

The crux of the argument is explained analytically by Levy et al. (2015b), and hinges on observing that many of the models from the literature use *linear* classifiers. Thus, consider a classifier which takes the concatenation of the vectors $\langle H, w \rangle$ learns a hyperplane \hat{p} to make its prediction. Then the hyperplane \hat{p} can also be viewed as a concatenation of two vectors:

$$\begin{aligned} & \hat{p}^\top \langle H, w \rangle \\ &= \langle \hat{H}, \hat{w} \rangle^\top \langle H, w \rangle \\ &= \hat{H}^\top H + \hat{w}^\top w \end{aligned}$$

This analysis shows that, when the hyperplane \hat{p} is evaluated on a novel pair, it lacks any form of direct interaction between H and w like the inner product $H^\top w$, but rather only learns to capture the notion of hypernymy through \hat{H} and \hat{w} , the *prototypicality vectors*. Without having some form of interaction, this Concat classifier has no way of estimating the relationship between the two

words. Furthermore, a linear classifier which uses the Diff vectors as input ($H - w$) will also have this flaw, since the hyperplane \hat{p} can be analyzed in this same fashion.

In their work, Levy et al. (2015b) back up this analysis with experimental evidence, showing that when the training/testing set is constructed to ensure that no lexical items are shared between the training and test sets (a variant of the experiments of Roller et al. (2014)), the performance of several classifiers, like Baroni et al. (2012) and Weeds et al. (2014), drop dramatically. Levy et al. (2015b) also propose a new model which incorporates the inner product term, which outperforms other models on several data sets. Interestingly, Asym does *not* suffer this fundamental flaw: although it uses the vector difference vectors as features, it also uses the *square-difference vectors* as input. Crucially, by the Law of Cosines, we can see that these square-difference features provide it these crucial inner product term:

$$\begin{aligned} & \sum_i (H_i - w_i)^2 \\ &= \sum_i H_i^2 + w_i^2 - 2(H_i w_i) \\ &= H^\top H + w^\top w - 2\mathbf{H}^\top \mathbf{w} \end{aligned}$$

This explains our observation in Section 3.1 that, without these square-difference terms, performance drops substantially.

Nonetheless, this raises a concern about what the difference terms $H - w$ actually provide. We propose a qualitative experiment which explains, in clear terms, why these terms are valuable, and leads to another model to extend this behavior. For simplicity, we focus our analysis on the linear Concat classifier, which exhibits the same behavior as Diff, but in a more obvious way.

In our qualitative experiment, we train a linear Concat classifier using syntactic distributional vectors on four separate data sets. We then analyze the trained models by comparing their hyperplanes to the *context vectors*. That is, we explicitly compare the \hat{H} vector to the syntactic context matrix C in Equation 2. This is a radically different view of than the prototypicality hypothesis of Levy et al. (2015b): rather than learning a prototype of hypernymy, our interpretation is that the Concat and Diff models learn to act as *feature detectors*, which identifies features (i.e. syntactic contexts), which are useful in identifying hypernymy. This interpretation and corresponding experiment is a one of our core contributions to the literature.

We train the model on four data sets: LEDS, BLESS, Medical, and TM14. LEDS and BLESS were also used in the Asym experiments, and are datasets covering hypernymy and non-hypernymy relations. Medical is a dataset of pairs of medical words and entailment labels, and was farmed using Information Extraction techniques (Levy et al., 2014). Finally, TM14 contains many varied word relations (like cause-effect, agent-object) which are annotated with entailment decisions by Turney and Mohammad (2015).

Table 5 shows the five contexts most similar to the hyperplane learned from each of the four datasets, and immediately explains why these models perform strongly. Nearly all of the contexts preferred by the model take the form of Hearst patterns (Hearst, 1992; Snow et al., 2004). The most recognizable and common pattern learned is the “such as” pattern, as in “animals such as cats”. These patterns have been well known to be indicative of hypernymy for over two decades.

LEDS	BLESS	Medical	TM14
nmod:such_as+animal	nmod:such_as+submarine	nmod:such_as+patch	amod+desire
acl:relcl+identifiable	nmod:such_as+ship	nmod:such_as+skin	amod+heighten
nmod:of ⁻¹ +determine	nmod:such_as+seal	nmod:including+skin	nsubj ⁻¹ +disparate
nmod:of ⁻¹ +categorisation	nmod:such_as+plane	nmod:such_as+tooth	nmod:such_as+honey
compound+many	nmod:such_as+rack	nmod:such_as+feather	nmod:with ⁻¹ +body

Table 5: Most similar contexts to the \hat{H} hyperplane learned by a Concat classifier.

Other interesting patterns are the “including” pattern (“animals including cats”) and “many” pattern (“many animals”). Although we list only the five most similar context items for the data sets, we find similar Hearst Pattern type contexts continue to dominate the list for the next 30-50 items.

Altogether, it is remarkable that the model identified these patterns using *only* distributional vectors and only the positive/negative example pairs. Since the model can be interpreted as a sort of *feature detector*, we call this model the H-feature Detector Model. We now show how these H-features can be improved using an iterative procedure similar to Principal Component Analysis.

3.3.2 The H-Feature Detector Model

Knowing that the Concat classifier acts primarily as a feature detector, we ask whether this can be combined with similarity-based insights of models like Asym. To this end, we propose a novel model which exploits the H-feature Detector model, extends its modeling power, and also adds in features for general similarity and distributional inclusion.

The model works through an iterative procedure similar to Principal Component Analysis (PCA). Each iteration repeatedly trains a Concat classifier under the assumption that it acts as a feature detector, and then explicitly *discards* this information from the distributional vectors. By training a new feature detector on these modified distributional vectors, we can find additional features indicative of entailment which were not captured by the first classifier. This is similar to how in Principal Component Analysis, the second principal component is computed after the first principal component has been accounted for.

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

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

Figure 3 gives a geometric illustration of the vector projection. If x forms the hypotenuse of a right triangle, $\text{proj}_{\hat{p}}(x)$ forms a leg of the triangle. This also gives rise to the *vector rejection*, which is the vector forming the third leg of the triangle. The vector rejection is orthogonal to the projection, and intuitively is “leftover” vector after the projection has been removed:

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

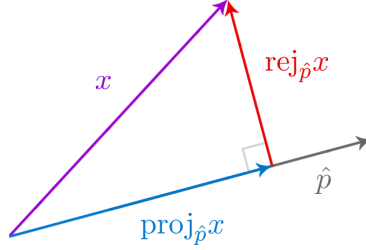


Figure 3: A vector \hat{p} is used to break H into two orthogonal components, its projection and the rejection over \hat{p} .

Using the vector rejection, we take a learned H-feature detector \hat{p} , and remove these features from each of the data points. That is, for every data point $\langle H, w \rangle$, we replace it by its vector rejection and rescale it to unit magnitude:

$$H' = \text{rej}_{\hat{p}}(H) / \|\text{rej}_{\hat{p}}(H)\|$$

$$w' = \text{rej}_{\hat{p}}(w) / \|\text{rej}_{\hat{p}}(w)\|$$

A new classifier trained on the $\langle H', w' \rangle$ data must learn a very different decision plane than \hat{p} , as \hat{p} is no longer present in any data points. This new classifier will perform strictly worse than the original, otherwise the first classifier would have learned this hyperplane. Nonetheless, it will be able to learn *new* H-features which the original classifier was unable to capture. By repeating this process several times, we can find several H-feature detectors, $\hat{p}_1, \dots, \hat{p}_n$.

In each iteration i of the procedure, we generate a four-valued feature vector F_i , based on the H-feature detector \hat{p}_i . Each feature vector contains (1) the similarity of H_i and w_i (before projection); (2) the H-feature detector \hat{p}_i applied to H_i ; (3) the H-feature detector \hat{p}_i applied to w_i ; and (4) the difference of 2 and 3.

$$F_i(\langle H_i, w_i \rangle, \hat{p}_i)$$

$$= \langle H_i^\top w ; H_i^\top \hat{p}_i ; w_i^\top \hat{p}_i ; H_i^\top \hat{p}_i - w_i^\top \hat{p}_i \rangle$$

These four “meta”-features capture all the benefits of the H-feature detector (slots 2 and 3), while addressing Concat’s issues with similarity arguments (slot 1) *and* distributional inclusion (slot 4).

The union of all the feature vectors F_1, \dots, F_n from repeated iteration form a $4n$ -dimensional feature vector which we use as input to another classifier. This classifier is trained on the exact same training data as each of the individual Hearst Pattern detectors, so the procedure only acts as a method of feature extraction. We use an SVM with an RBF-kernel, as we found it to work best, though several nonlinear classifiers also perform well.

We compare our H-feature detector model to several existing and alternative baselines from the literature. Namely, we include a baseline Cosine classifier, which only learns a threshold which maximizes F1 score on the training set; three linear models of prior work, Concat, Diff and Asym; and the RBF and Ksim models found to be successful in Kruszewski et al. (2015) and Levy et al. (2015b) respectively. We also include Asym + Concat, which was used in Beltagy et al. (2016).

Model	LEDS	BLESS	Medical	TM14
Linear Models				
Cosine only (Baseline)	.787	.208	.168	.676
Concat	.794	.612	.218	.693
Diff (Weeds et al., 2014)	.805	.440	.195	.665
Asym (Roller et al., 2014)	.865	.510	.210	.671
Asym + Concat (Beltagy et al., 2016)	.843	.631	.240	.701
Nonlinear Models				
RBF	.779	.574	.215	.705
Ksim (Levy et al., 2014)	.893	.488	.224	.707
H-Feature Detector (Roller and Erk, 2016b)	.901	.631	.260	.697

Table 6: Mean F1 scores for each model and data set.

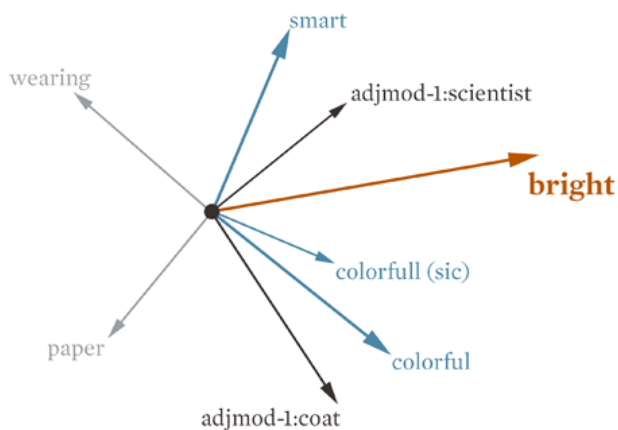


Figure 4: Caption here.

We cannot include a additional comparisons like Ksim+Asym, because Ksim is based on a custom SVM kernel which is not amenable to combinations.

Table 6 the results across all four data sets for all of the listed models. Our H-Feature model improves significantly² over Concat in the LEDES, BLESS and Medical data sets, indicating the benefits of combining these the aspects of similarity and distributional inclusion with the H-feature detectors of Concat. The Asym + Concat classifier also improves over the Concat baseline, further emphasizing these benefits. Our H-feature model performs approximately the same as Ksim on the LEDES and TM14 data sets (no significant difference), while significantly outperforming it on BLESS and Medical data sets.

3.4 Lexical Substitution (Roller and Erk, 2016a)

In this last section of our work, we switch over to our efforts on the Lexical Substitution (LexSub) task. In the Lexical Substitution task, we are given a sentence and target word, and must choose possible substitutes from the entire vocabulary which preserve the meaning of the sentence. As described in Section 2.4, we view LexSub as a proxy which captures a wide degree of lexical entailments *in context*.

We propose a new measure, which extends a previously successful, simple model to estimate the appropriateness of a lexical substitute. As a introduction point for our measure, we review the work of Melamud et al. (2015b), which introduced several unsupervised measures for word similarity in context: namely *balAddCos* and *addCos*. The main insight of these unsupervised measures is the use of *context vectors*, related to the insights discussed in Section 3.3.1. Crucially, the models of Melamud et al. (2015b) depend on syntactic context vectors, which have been found more successful than just BoW measures in the literature (Erk and Padó, 2008; Dinu and Lapata, 2010; Thater et al., 2010; Van de Cruys et al., 2011). We note that these models are *not* the state of the art, (Melamud et al., 2015a; Melamud et al., 2016) but perform competitively while remaining simple, extensible, and highly interpretable.

In the work of Melamud et al. (2015b) and others, substitutes are modeled using a mixture of *out-of-context similarity* and *in-context appropriateness*. The out-of-context similarity uses standard distributional semantics cosine similarity (Equation 1) in order to estimate how similar the target is to the substitute, and remains fixed regardless of the sentential context. For this reason, this out-of-context (OOC) similarity is also used as a baseline in the literature.

The in-context appropriateness attempts to directly model whether a proposed substitute fits in the given sentential context. That is, if one replaces the target with the substitute directly, would a reader still consider this word *selectionally* agrees in the sentence. For example, the in-context measures will give a low score for *dog purrs*, since dogs usually do not purr. To this end, it assumes the sentence is parsed, so that we have the full set of syntactic neighbors of the target, C . Each of the context vectors corresponding to elements of C are then evaluated for their fit with the proposed substitute, as illustrated in Figure 2. For a given target t , substitute s and context C , the final *addCos* score is given as

$$\text{addCos}(s|t, C) = \text{cosine}(s, t) + \sum_{c \in C} \text{cosine}(s, c). \quad (3)$$

The model can be intuitively understood using the diagram in Figure 4. For a given target “bright,” we will choose the substitute (“smart” or “colorful”) which is *closer* to the given context. If the context is “scientist” as in “bright scientist,” we will shift our prediction away from “colorful” and closer to “smart,” and vice versa for “bright coat.” This remains one of the simplest successful models of Lexical Substitution to date. Melamud et al. (2015b) also consider variants of this measure, including *balAddCos*, which equally weights the out-of-context similarity with the in-context appropriateness:

$$\text{balAddCos}(s|t, C) = \text{cosine}(s, t) + \frac{1}{|C|} \sum_{c \in C} \text{cosine}(s, c). \quad (4)$$

²Bootstrap test, $p < .01$.

In our work, we propose a new measure, called Probability-in-Context (PIC),³ which estimates the appropriateness of a substitute in a given context (Roller and Erk, 2016a). Similar to *balAddCos*, the measure has two equally-weighted, independent components measuring the appropriateness of the substitute for both the target and the context, each taking the form of a softmax:

$$\begin{aligned} \text{PIC}(s|t, C) &\propto P(s|t) \times P(s|C) \\ P(s|t) &= \frac{1}{Z_t} \exp \{s^\top t\} \\ P(s|C) &= \frac{1}{Z_C} \exp \left\{ \sum_{c \in C} s^\top [Wc + b] \right\}, \end{aligned} \tag{5}$$

where the Z_t and Z_C are normalizing constants. Our model differs from *balAddCos* in one major way: we base our similarity estimates using the *unnormalized* inner product $s^\top t$ and $s^\top c$, rather than normalized cosine similarities. We also introduce two additional parameters, W and b , which act as a simple linear transformation over the original context vectors. These parameters are learned from the original corpus, and serve only to tune how the fixed distributional vectors act in this alternative objective function.

To identify the contribution of this parameterization versus the softmax objective, we also introduce to a non-parameterized PIC (*nPIC*), which does not contain the extra parameters:

$$\begin{aligned} \text{nPIC}(s|t, C) &= P(s|t) \times P_n(s|C) \\ P_n(s|C) &= \frac{1}{Z_n} \exp \left\{ \sum_{c \in C} s^\top c \right\} \end{aligned} \tag{6}$$

We compare our model to that of an out-of-context baseline (cosine) and the *addCos* and *balAddCos* models of Melamud et al. (2015b), which outperformed other prior work at the time of its publication. We compare the models on three data sets: SE07, the original LexSub data set (McCarthy and Navigli, 2007) which was explicitly developed to capture polysemy; Coinco, a recent LexSub data set which contains substitutes for all content words in a small corpus (Kremer et al., 2014); and TWSI2, which was developed to be a large collection of lexical substitutes from a diverse corpus (Biemann, 2012). We measure performance in Mean Precision@1, which measures whether our best proposed substitute is contained within the set of gold substitutes provided by annotators. In all models, we exclude any words sharing the same lemma as the target, e.g. if the target is “barking” we do not propose “bark.”

Table 7 contains results for all measures across all datasets. We observe that PIC outperforms all other models by a significant margin,⁴ including a relative 30% improvement over *balAddCos* in SE07 and Coinco. The nPIC also improves substantially over the other baselines, indicating we gain benefit both from the new objective function and the additional parameterization. We next strive to understand why both measures have a clear improvements over the baseline models.

We characterize the models using a cherry-picked example, given in Table 8. Although this example does not perfectly illustrate the Lexical Substitution task, it does well at giving intuition as

³PIC is not strictly a probability measure; the name is a backronym for the purpose of the paper’s title.

⁴Wilcoxon signed-rank test, $p < 0.01$

Measure	SE07	Coinco	TWSI2
Cosine (OOB Baseline)	11.7	10.9	9.8
addCos	12.9	10.5	7.9
balAddCos	13.4	11.8	9.8
nPic	17.3	16.3	11.1
PIC	19.7	18.2	13.7

Table 7: Lexical Substitution results for the all-words prediction task, measured in Mean Precision@1.

OOB	<i>balAddCos</i>	<i>nPIC</i>	<i>PIC</i>
You can sort of challenge them well, did you really know the time when you said yes?			
trully	proably	realy	actually
actually	trully	truly	truly
actaully	acutally	actually	already
acutally	actaully	hardly	barely
proably	probaly	definitely	just

Table 8: Example where the *PIC* performs strictly better than other models. The target word and correct answers are bolded.

to why our models perform better. We see that all four measures tend to pick excellent substitutes which *semantically* agree with the original target. However, the cosine and *balAddCos* models have a large number of misspelled words in their list, while the *nPIC* and *PIC* measures contain mostly correct spellings. This is because, somewhat surprisingly, the *length* of the distributional vectors correlates strongly with the *unigram* statistics of the word (Wilson and Schakel, 2015). Therefore, by using the unnormalized inner product, rather than cosine, our model naturally incorporates unigram priors, allowing it to downweight rare, similar words. Indeed, a quantitative analysis of the *W* and *b* parameters finds that they additionally exaggerate this unigram bias (Roller and Erk, 2016a). Intuitively, it seems natural that unigram biases should hold a strong role in Lexical Substitution, and that our model should wish to exploit this information.

4 Proposed Work

We now describe the proposed methods of further research. The proposed future work breaks into two broad categories: short-term proposals, which must be completed for the final thesis, and long-term proposals, which are more ambitious research directions that may take much longer to pursue successfully.

The proposed short term work focuses predominantly on how the more recent successful re-

search may contribute to a larger RTE system: While the completed work has shown successful results on the two tasks of Lexical Entailment and Lexical Substitution, these newer models have not yet been applied to the end goal of Textual Entailment. We propose multiple methods to test our models in an end-to-end RTE system.

The long term work follows from three broad directions forward: (1) encouraging ontological consistency of predictions in Lexical Entailment; (2) better integration of a wider context into the Lexical Substitution system; and (3) a more sophisticated distributional model which reuses information when possible.

4.1 Short Term Proposals

In this section, we propose and discuss several additional experiments which should be completed for the final thesis. We break the section into experiments for Lexical Entailment, and experiments for Lexical Substitution. All are proposed primarily with integration into a final RTE system in mind.

4.1.1 Lexical Entailment

H-features for non-Hypernymy Relations In Section 3.3, we discussed how certain distributional models act as *H-feature* detectors, which identify contexts highly indicative of hypernymy, and discuss our model which exploits multiple H-feature detectors in order to improve the modeling power of a hypernymy detection system, and improves results over comparable models.

However, there are many relations *other* than hypernymy which are useful in considering textual entailment: for example, identifying co-hyponymy is useful as a negative signal for entailment, and identifying meronymy is critical to our motivating example in the Introduction. Indeed, the results shown in Table 1 show the accuracy of the Asym and Baroni classifiers on a *four-way* relationship prediction task: hypernymy, co-hyponymy, meronymy, and random, but the experiments in Table 6 only describe performance in a binary hypernymy-or-not classification task. We propose to extend and evaluate the H-features model of Section 3.3 to handle non-hypernymy relations. We believe the model’s performance can be improved by better modeling the non-hypernymy cases, and that the model will additionally discover H-features indicative of other relations.

There are several ways that the model could be extended. The one we believe will be most successful is one that trains several binary H-features models: one for hypernymy-vs-non-hypernymy, one for meronymy-vs-non-meronymy, etc. Similar to how the PCA procedure was used only as a form of feature-extraction for the final prediction, each of the binary classifier iterations will be also used for feature extraction for a final classifier. That is, we will use the procedure described in Section 3.3 to extract several iterations of features for hypernymy, then completely repeat the procedure for meronymy and so forth. The resulting features from each of the classifiers will be concatenated for a final four-way classifier prediction. Another alternative would be to try to learn the four-way classifiers concurrently (e.g., a softmax instead of logistic regression), and extract the corresponding H-features at this level.

There are interesting research questions that stem from this procedure, beyond just final performance scores. One is what distributional-level features will be learned as prototypical of meronyms,

or co-hyponyms? As we saw in Table 5, the classifier automatically learned to pick out well-known Hearst patterns, indicative of hypernymy. It remains to be seen whether it will pick out additional Hearst patterns indicative of other relations: for example, simple conjunctions for co-hyponymy (e.g., *cats and dogs*) or the possessive for meronymy (e.g., *cat's tail*).

H-features Model for RTE The results of Section 3.2 showed that the Asym model can provide improvements in a complete, end-to-end RTE system, especially when combined with traditionally hand-engineered features. However, we have not yet considered whether our new H-features model also contributes to an end-to-end RTE pipeline. We suspect, given its substantial improvements over Asym in lexical datasets, that it will also be able to improve the end-to-end system.

It remains unclear what is the best way to integrate it with the additional hand-engineered features used in the pipeline. One possibility would be to simply use the H-feature detector procedure to provide additional information to the lexical entailment classifier, but there are complications in that the H-feature detector demands nonlinear models, while the hand-engineered features prefer linear classifiers. We suspect this may be alleviated with careful hyperparameter tuning.

These proposed experiments will also need to be tightly coupled with the complete non-hypernymy detector discussed in the previous section. If H-features are useful for predicting co-hyponymy or meronymy, it also stands to reason that they should be useful in the final RTE subsystem. We will need to explore the best form of combination with both kept in mind.

4.1.2 Lexical Substitution in RTE

In Section 2.4, we argued that Lexical Substitution is related to textual entailment, partially standing as a kind of in-context entailment. We have yet to provide any results indicating that Lexical Substitution contributes in an RTE system. We propose that the Lexical Substitution model described in Section 3.4 be used as additional features in the Lexical Entailment classifier described in Section 3.2. Although we hope that our Lexical Substitution model can positively contribute to the RTE pipeline, we suspect improvements may be marginal.

The obvious way our Lexical Substitution model could contribute to the task is by simply including the context-similarity features of the Lexical Substitution model as another feature in the Lexical Entailment classifier. That is, for a pair of words in the SICK dataset, measure the similarity of their contexts using the $P(s|C)$ or $P_n(s|C)$ values proposed in Equations 5 and 6. Although polysemy is not a significant issue in the SICK dataset, we hope the additional information about context will bolster predictions in borderline cases.

It may be necessary to find additional tricks in order to best integrate this information into the lexical entailment classifier. For example, we found the cosine binning trick described in Section 3.2 was necessary for getting distributional similarity to contribute. Similar tricks, like binning or isolating the components of PIC, are likely necessary in order to see any positive contribution.

Ultimately though, we suspect improvements to RTE may be marginal, or even negative, as polysemy is uncommon in the SICK dataset. We may need to analyze differences in classification prediction, and which examples are *most* affected by the LexSub features, in order to properly characterize contribution.

4.2 Long Term Proposals

We now describe some possible longer term research questions which could be addressed in the final thesis. Success in any of these items would be significant contributions to the field, and are therefore more ambitious and risky than the short-term proposals.

4.2.1 Ontology Constraints in Hypernymy Prediction

Presently in our hypernymy-prediction models, relationships between all pairs of words are made independently: the prediction of whether “cat is an animal” has no bearing on whether “dog is an animal,” aside from their distributional similarity. While this is a straightforward application of machine learning principles to the problem, it ignores an important fact: that hypernymy is just one aspect of a complete, well-structured ontology. Yet, since we predict each word pair individually, there is no guarantee the output of the system over all pairs will also be well-structured.

For example, a hypernymy prediction model could predict that both “animal is a hypernym of dog” and that “dog is a hypernym of animal,” even though we know hypernymy is non-reflexive. Or it could predict that “a golden retriever is a dog” and “dog is an animal,” but incorrectly predict that “gold retriever is *not* an animal,” violating the transitive property of hypernymy. These properties of hypernymy are inherent to its definition, and our models should be take this into account.

With this in mind, is it possible to modify our models such that such ontological constraints are guaranteed or preferred? One possibility would be to use the *confidence scores* associated with the hypernymy predictions, and simply revise the least-confident predictions to conform to constraints in a post-processing step. For example, in our reflexive example above, the more confident of the two predictions will be assumed to be the correct one.

This idea could likely benefit further from our short-term proposal to see how well the H-features model does at predicting relations other than hypernymy: for example, if we are highly confident that two words are co-hyponyms, then we can become more confident that they share a common hypernym, and vice versa.

This idea of enforcing ontological constraints is not new: it was previously explored by Caraballo (1999), who use a clustering algorithm in order to find co-hyponym terms, and then predict hypernyms using the entire clusters. It was also examined some in Snow et al. (2004), who used syntactic Hearst Patterns to make predictions about hypernymy, and then linearly interpolated the prediction confidences in order to better conform to these hard constraints. Later, Snow et al. (2006) proposed a probabilistic model over *ontologies*, and an algorithm for searching over entire ontologies constrained by rules about the transitivity of hypernymy, and reflexivity of co-hyponymy. This ontology search is then used in order to find the single ontology which has maximal probability according to the evidence provided by Hearst patterns, while also not violating any of the hard constraints.

Although Snow et al. (2006) found their ontology searching algorithm highly successful with the use of the lexico-syntactic patterns indicative of hypernymy, such an approach is yet to be tried on classifiers which make use of *distributional information* about words. Therefore, the most reasonable first course of action would to reimplement and examine whether their model is compatible with the distributional models of hypernymy prediction. However, their model supports

only two forms of constraints (transitivity of hypernymy and reflexivity of co-hyponyms), leaving open questions about how other constraints should be imposed.

Another possibility is through the use of the same technology powering the end-to-end RTE system of Beltagy et al. (2016): Markov Logic Networks (MLNs). Markov Logic Networks provide a framework for doing probabilistic logical inference over sets of weighted atoms and logical rules. MLNs are given a set of weighted First Order Logic (FOL) rules and a database of atoms, and give reweighted predictions about the probabilities of atoms based on their consistency with the logical rules. MLNs can encode many different relations as rules, and perform joint updating of *all* lexical relationship predictions. For example, the rules for transitivity and reflexivity discussed in Snow et al. (2006) could be encoded as:

$$\begin{aligned} \forall x, y, z. \text{hyper}(x, y) \wedge \text{hyper}(y, z) &\rightarrow \text{hyper}(x, z), \\ \forall x, y. \text{cohyp}(x, y) &\leftrightarrow \text{cohyp}(y, x), \end{aligned}$$

but other rules, such that co-hyponyms share a hypernym, may also be encoded:

$$\forall x, y, z. \text{cohyp}(x, y) \wedge \text{hyper}(x, z) \rightarrow \text{hyper}(y, z).$$

MLNs ability to incorporate *weighted rules* would also give room for flexibility in constraint importance, and allow for *some* violations of constraints when the evidence is simply overwhelming. Therefore, we believe both the model of Snow et al. (2006) and an MLN-based model to be strong candidates for improving lexical relationship prediction by enforcing ontology constraints.

4.2.2 Sophisticated Contexts for Lexical Substitution

Wider Contexts in Lexical Substitution In the model of Lexical Substitution discussed in Section 3.4, we showed how the syntactic neighbors of a target word are useful in predicting what are the lexical substitutes, or in-context synonyms, of the target word. Although the syntactic neighbors can indeed capture some kinds of *long-distance dependencies*, there is much greater deal of context available which is presently not used by the model: the *entire rest of the sentence*.

Consider if our PIC model were asked to find the best verb to fill-in-the-blank in the following two simple sentences:

The dog ___ the tennis ball.
The dog ___ the meat ball.

In both cases, PIC will be given the exact same context for the missing verb: its subject should be *dog* and its object should be *ball*. However, humans know that the dog is more likely to *chase* or *fetch* the tennis ball, while it is more likely to *eat* the meat ball. Without being provided the additional information about the ball, the model has absolutely no way to distinguish these two cases. How can this information be integrated into our model?

Even beyond this simple example, it is already clear from prior work that additional context can be very useful in the task. Dinu and Lapata (2010) considers a probabilistic model based on a wide Bag-of-Words context, and Van de Cruys et al. (2011) propose a model which combines syntactic

distributional space with a bag-of-words distributional space, showing modest improvements over each individual space. Additionally, Kawakami and Dyer (2016) obtained state-of-the-art results on Lexical Substitution using a neural network language model which encodes the *entire sentence* as input, though their model also depends heavily on the use of machine translation data. It is clear that there is more useful information available than our own Lexical Substitution model is provided.

There two distinct ways we could implement wider contexts into our Lexical Substitution model: using linguistic knowledge, and using neural networks. In the former, we could simply use existing linguistic knowledge in order to model additional syntactic patterns from large corpora. That is, in addition to modeling the direct syntactic neighbors of words, we could also add pattern-based rules for modeling indirect neighbors. For example, we could introduce an additional context in our distributional space for `dobj+compound+_+meat`, marking that the direct object the verb is compounded with “meat,” and so on for other nouns. Since our model uses collapsed prepositional phrases, this is already partially implemented (e.g., we already model “to the store” as `prep:to_store` rather than just `prep:to`).

A variation of this approach was discussed in Padó and Lapata (2007), the original proposal of syntactic distributional spaces. In their model, they also extracted contexts for dependency chains two and three hops away from the target word, rather than the only the direct neighbors. Since then, most models have focused mostly on direct neighbors, since longer chains substantially increase complexity, sparsity, and model parameters. If we use this linguistic approach, issues of scaling and sparsity will likely plague our model.

The other possibility for this problem would be to employ modern neural network models, like the Long Short Term Memory (LSTM) model (Hochreiter and Schmidhuber, 1997). The LSTM model has become extremely popular in the field of Natural Language Processing, thanks to recent advances in network initialization (Glorot and Bengio, 2010), training techniques (Duchi et al., 2011; Kingma and Ba, 2014), and hardware like Graphics Processing Units (GPUs). Indeed, LSTMs are the basis for the successful model of Kawakami and Dyer (2016) mentioned previously. LSTMs are a form of a Recurrent Neural Network, which takes as input a variable-length sequence of items (tokens), and make some prediction. They have been successfully applied countless areas of NLP, including language modeling (Sundermeyer et al., 2012; Jozefowicz et al., 2016), sentiment analysis (Kumar et al., 2016), machine translation (Sutskever et al., 2014), textual entailment (Bowman et al., 2015), question answering (Hermann et al., 2015), and recently even lexical entailment (Shwartz et al., 2016).

More concretely, we could integrate wider syntax using an approach similar to that of Shwartz et al. (2016). In their model, a syntactic chains connecting two target words was used to classify whether the words stand in a hypernymy relationship or not, acting as a modern neural-network version of Snow et al. (2004). We propose using similar syntactic chains to simply predict the *last word* in the chain, training using billions of syntactic chains extracted from large corpora. This model would have the capability of capturing and remembering relevant long-distance information whenever it is helpful in predicting the final word. One challenge in applying this model is scaling it, as LSTM models can take days or weeks to train when their final output is a prediction over the entire vocabulary.

Joint Models of Syntactic Neighbors Another issue with the lexical substitution model discussed in Section 3.4 is that the model fundamentally assumes independence between each of the syntactic neighbors. Ultimately, the model suggests substitutes in the above examples by asking, “What does a dog do? What is done to a ball? What is the intersection of these two sets?” Our use of unnormalized inner products in Equation 5 means that one question may have more weight than another, but they are still considered independently: the question is never “What does a dog do to a ball?”

It is worth considering whether this independence assumption can be relaxed in some way. Intuitively, it seems obvious that a joint model should be helpful. Yet, one major issue is that the number of syntactic neighbors is variable: consider that a verb may have attachments may have only one attachment (intransitive), two attachments (transitive), or more (ditransitive, prepositional phrase attachments, adverbs, etc). Similarly, nouns can also stand in many syntactic relations simultaneously (compound, adjective, prepositional phrase, verb relation, and others). Since the number of attachments is variable, it is difficult to define a probabilistic model which does not demand at least *some* independence assumptions. Even if we were to define the model over all possible syntactic relations for every word, the issue would not be solved: consider a “small, furry, brown mouse,” which has three modifiers all standing in the `adjmod` relation.

Even if we could define such a model, it would likely be plagued the typical extreme sparsity issues: for many sentences in our dataset, an *exact* combination of attachments seen is unlikely to appear anywhere, even in extremely large corpora. Therefore, a naive joint model is likely to estimate the probability as zero for most everything.

As with the proposed methods for including wider contexts, we could potentially address these issues using either linguistic knowledge, or neural networks. To address the concerns using linguistic knowledge, we could again mark certain rules should be modeled jointly, and assume independence between the remaining syntactic relations. For example, we could model transitive verbs jointly, but assume independence from adverbs and prepositional modifiers; or we could jointly model two adjective modifiers, but assume independence once we encounter three or more. Again, rules similar to the ones discussed in Padó and Lapata (2007) would be a good starting point, and others could be proposed in qualitative analysis of the data.

The other possibility is to use LSTMs in order to jointly model the different modifiers. Since LSTMs are able to handle a variable-length sequence as input, they seem to be a good candidate for creating a joint model over all the attachments in a principled manner. Unfortunately, LSTMs also assume that the *order* of the sequence is important, which is not the case for our problem: there is no inherent ordering of syntactic attachments. We could canonicalize the ordering (simply insisting that, for example, subjects are always presented before objects), but it remains unclear whether this is useful. We could also *randomize* the order the attachments are presented: since there are many permutations, this could substantially increase the amount of training data available. However, preliminary experiments showed difficulty with this second approach: the PIC model came about when our first attempts at using LSTMs were unsuccessful.

Nonetheless, it is encouraging that the same linguistic and neural network approaches could be potentially useful for both introducing wider context, and joint modeling.

4.2.3 Generalized Distributional Spaces for Entailment and Substitution

Our final long term proposal is fairly different from the other two: while those were ideas to directly impact performance on the Lexical Entailment and Lexical Substitution, our last idea focuses more broadly on how we can improve distributional spaces altogether.

We begin with an observation about the process described in Section 2.2 when constructing a syntactic distributional space. During construction, the final step involves finding a matrix factorization for the extremely large, extremely sparse word-context co-occurrence matrix. In syntactic spaces, the contexts are distinguished by the neighboring word *together* with its syntactic relation. That is there is one context for `nsubj_dog` and another for `dobj_dog`. This is a powerful notion, and one that enables the syntactic spaces their ability to model selectional preferences.

Yet, modeling each of these contexts independently also seems to be wasting a large amount of information: that the same word is being modified by two different syntactic relations. When this is ignored, we assume we have excellent statistics for how every word stands in every relation, but this is unrealistic, and one of the important steps in constructing a syntactic distributional space is to limit modeling to only the most frequent contexts. But in defining a cutoff at all, we are also omitting a significant amount of data. Do we really want to throw away all this data?

We ask whether the *syntactic relations* be separated from the *words* in syntactic distributional spaces? Can these context vectors we rely heavily on in our work be *generatively modeled*? One way to approach this would be to model it as a composition function:

$$\text{nsubj}_{\vec{j}}\text{-dog} = f(\text{nsubj}, \vec{dog}).$$

In this view, we would need to learn one function, f , which takes in a syntactic relation and a word, and produces a *context vector* on how `dog` acts in the subject position. One possibility would be to use vector concatenation, treating the relation and the word as separate components, independent of each other. Intuitively, this does not seem to capture the the essence of our proposal, but it could act as a baseline model:

$$\text{nsubj}_{\vec{j}}\text{-dog} = \langle \vec{dog}; \vec{\text{nsubj}} \rangle.$$

Another possibility would be to treat the procedure as a form of post-processing generalization. For example, we could factorize the context vectors as we do now, and then attempt to learn to predict the observed context vectors from the input components. For example, perhaps after performing the concatenation, a simple linear regression could combine them:

$$\text{nsubj}_{\vec{j}}\text{-dog} = W \langle \vec{dog}; \vec{\text{nsubj}} \rangle + \vec{b}.$$

This is a basic model, but it would still enable us to predict information about novel combinations of syntactic relations and words. One could increase the representational power using a more sophisticated regression model, like neural networks.

Another variation of this linear regression idea, would be to model each syntactic relation as a function *applied* to the vector. For example, we would learn a separate transformation for `nsubj`, `dobj`, etc:

$$\text{nsubj}_{\vec{j}}\text{-dog} = W_{\text{nsubj}} \vec{dog}$$

This gives the model many more free parameters, but may present some difficulty for the rarest syntactic relations, where there could easily be more parameters in W than examples, though perhaps heavy regularization of W , or even constraining it to be diagonal could be beneficial. Modeling the behavior in this way would draw parallels to other areas of the literature, like compositional distributional semantics (Baroni and Zamparelli, 2010; Coecke et al., 2011; Grefenstette and Sadrzadeh, 2011; Baroni et al., 2014a), the bilinear models proposed in GloVe (Pennington et al., 2014), and the recursive composition models for sentiment analysis (Socher et al., 2013b).

Alternatively, we could try to learn the distributional vectors from scratch using an alternative factorization. Recall that the dimensionality reduction procedure simply tries to predict the Pointwise Mutual Information between a word and a context. Rather than performing the dimensionality reduction, then regression, we could encode our generative principals into the dimensionality reduction. In this way, a successful model may look something more like *tensor factorization*. For example, we could model the PMI between a word (v), and a relation-context pair (r, c) as:

$$PMI(v, r, c) = \mathbf{v}^\top \mathbf{W}_r \mathbf{c},$$

where \mathbf{W}_r is a matrix. This is essentially the same as the linear regression model proposed in the above paragraph, but performed at a different step. Unfortunately, tensor factorization is generally very difficult (Håstad, 1990; Hillar and Lim, 2013). Finding the ideal factorization for our problem is likely a very difficult optimization problem without major simplifying assumptions, and beyond our own expertise.

However, an excellent place to start would be in the Information Extraction literature, especially those focused on Statistical Relational Learning. These works seek to find generative representations for (subject, verb, object) triples, like `BornIn(John, Athens)`, and there is a rich literature regarding approaches to this very difficult problem. The approach of Nickel et al. (2011) inspired the model described in the previous paragraph, but many other models have been proposed (Socher et al., 2013a; Riedel et al., 2013; Yang et al., 2014; Kuleshov et al., 2015; Trouillon et al., 2016).

Regardless, if we assume that we *can* find any model which successfully predicts context vectors in a generative manner, then it could lead to substantial improvements on both tasks of Lexical Entailment *and* Lexical Substitution. For example, we saw in Table 5 that certain syntactic relations are indicative of hypernymy. Therefore, being able to plug in *any* two words to and estimate the likelihood that they will stand in this syntactic relation, like `nmod:such_as`. The problem would contribute even more greatly to the Lexical Substitution model we presented: in a preliminary examination, we found that some 18% of the Coinco dataset contains at least one context not available in our distributional space. The ability to *generate* representations for these contexts would be a simple way to provide the model with more disambiguating information, which should help in the end task.

Indeed, as we described above, a novel model could easily have larger implications in other areas of Natural Language Processing, and is therefore probably the most ambitious and difficult of our long term proposals.

5 Conclusion

Distributional Semantics has come a long way in its ability to contribute to difficult Natural Language Processing tasks. In this proposal, we several methods for distilling *lexical knowledge* out of distributional vectors. We discussed two models of hypernymy detection, which take as input pairs of words, and predict whether those models stand in a hypernymy relation. Through detailed analysis, we have a clear picture of how these models work, and how they relate to some of the linguistic literature on Lexical Entailment, like the Distributional Inclusion Hypothesis and Hearst Patterns. We also saw that at least one of these models is able to positively contribute to an RTE system We discussed a new model of Lexical Substitution, which we argue is related to lexical entailment by acting as a form of *in-context synonym prediction*. Our Lexical Substitution model outperforms some comparable baselines, and analysis shows its improvements derive predominantly from exploiting simple unigram priors.

We also discussed multiple directions for how our work could be extended in the future. In the short term, we focus on exploring how our H-features model could also contribute to the RTE Lexical Entailment classifier, and whether the H-features model is able to model relations other than hypernymy. We also discussed several ways in which the Lexical Substitution model could contribute to the RTE Lexical Entailment Classifier. Progress in these areas would contribute significantly to our thesis that our models extract useful lexical information for Recognizing Textual Entailment.

Finally, we discussed different long-term directions for future research. These ideas are much larger, and less guaranteed than the short-term research proposals, but also reflect our opinion on where the field should go from here. Namely, we considered how Hypernymy and Relationship prediction could be improved by imposing simple constraints inherent to ontologies. We also discussed several ways in which more context could be exploited in Lexical Substitution, or how we could remove some of the independence assumptions made by our current model. We note that improvements could come by either exploiting linguistic knowledge about how different syntactic relations interact, or by using recent successful neural network techniques. Lastly, we considered how current syntactic distributional spaces make an inefficient use of the available information, and discussed ways they could be extended to be *generative* over syntactic relations, instead of just artificially limiting the statistics they model. Improvements in these spaces would likely improve performance of distributional models in many tasks, including Lexical Entailment and Lexical Substitution.

References

- Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 86–90. Association for Computational Linguistics.
- Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.
- Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10, Edinburgh, UK.
- Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.
- Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 2012 Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32, Avignon, France.
- Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. 2014a. Frege in space: A program for compositional distributional semantics. *Linguistic Issues in Language Technology*, 9(6):5–110.
- Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014b. Don’t count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 238–247, Baltimore, Maryland, June. Association for Computational Linguistics.
- I. Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets markov: Deep semantics with probabilistic logical form. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, pages 11–21.
- I. Beltagy, Stephen Roller, Gemma Boleda, Katrin Erk, and Raymond Mooney. 2014. UTEXAS: Natural language semantics using distributional semantics and probabilistic logic. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 796–801, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.
- I. Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond Mooney. 2016. Representing meaning with a combination of logical and distributional models. *Special Issue of Computational Linguistics on Formal Distributional Semantics*.
- Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth pascal recognizing textual entailment challenge. *Proceedings of the Text Analytics Conference*, 9:14–24.
- Chris Biemann. 2012. Turk bootstrap word sense inventory 2.0: A large-scale resource for lexical substitution. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, pages 4038–4042, Istanbul, Turkey, May. European Language Resources Association.
- Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. 2014. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 642–646, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

- David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.
- Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Proceedings of Semantics in Text Processing*.
- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September. Association for Computational Linguistics.
- John A Bullinaria and Joseph P Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*, 39(3):510–526.
- Sharon A. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 120–126, College Park, Maryland, USA, June. Association for Computational Linguistics.
- Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.
- Daoud Clarke. 2009. Context-theoretic semantics for natural language: an overview. In *Proceedings of the 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 112–119, Athens, Greece, March. Association for Computational Linguistics.
- Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2011. Mathematical foundations for a compositional distributed model of meaning. *Linguistic Analysis*, 36(1-4):345–384.
- Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.
- Ido Dagan, Oren Glickman, and Bernardo Magnini, 2006. *The PASCAL Recognising Textual Entailment Challenge*, pages 177–190. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.
- Scott Deerwester, Susan Dumais, Thomas Landauer, George Furnas, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–407.
- Georgiana Dinu and Mirella Lapata. 2010. Measuring distributional similarity in context. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1162–1172, Cambridge, MA, October. Association for Computational Linguistics.
- John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.
- Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 897–906, Honolulu, Hawaii, October. Association for Computational Linguistics.
- Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.
- John R. Firth. 1957. A synopsis of linguistic theory 1930–1955. In *Studies in linguistic analysis*, pages 1–32. Blackwell Publishers, Oxford, England.
- Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 2014 Annual Meeting of the Association for Computational Linguistics*, pages 1199–1209, Baltimore, Maryland.

- Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics.
- Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 2010 International Conference on Artificial Intelligence and Statistics*, volume 9, pages 249–256.
- Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.
- Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.
- Johan Håstad. 1990. Tensor rank is np-complete. *Journal of Algorithms*, 11(4):644–654.
- Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 1992 Conference on Computational Linguistics*, pages 539–545, Nantes, France.
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.
- Christopher J Hillar and Lek-Heng Lim. 2013. Most tensor problems are np-hard. *Journal of the ACM*, 60(6):45.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.
- David A Jurgens, Peter D Turney, Saif M Mohammad, and Keith J Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 356–364. Association for Computational Linguistics.
- Kazuya Kawakami and Chris Dyer. 2016. Learning to represent words in context with multilingual supervision. In *Proceedings of 2016 International Conference on Learning Representations*.
- Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16:359–389, 10.
- Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What substitutes tell us - analysis of an “all-words” lexical substitution corpus. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 540–549, Gothenburg, Sweden, April. Association for Computational Linguistics.

- Germán Kruszewski, Denis Paperno, and Marco Baroni. 2015. Deriving boolean structures from distributional vectors. *Transactions of the Association for Computational Linguistics*, 3:375–388.
- Volodymyr Kuleshov, Arun Tejasvi Chaganty, and Percy Liang. 2015. Tensor factorization via matrix factorization. In *Proceedings of the 2015 International Conference on Artificial Intelligence and Statistics*.
- Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *The 33rd International Conference on Machine Learning*.
- Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 329–334, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.
- Thomas K Landauer and Susan T Dumais. 1997. A solution to plato’s problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.
- Daniel D. Lee and H. Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, pages 556–562. MIT Press.
- Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *The First Joint Conference on Lexical and Computational Semantics*, pages 75–79, Montréal, Canada, June. Association for Computational Linguistics.
- Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 302–308, Baltimore, Maryland, June. Association for Computational Linguistics.
- Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.
- Omer Levy, Ido Dagan, and Jacob Goldberger. 2014. Focused entailment graphs for open ie propositions. In *Proceedings of the 2014 Conference on Computational Natural Language Learning*, pages 87–97, Ann Arbor, Michigan.
- Omer Levy, Yoav Goldberg, and Ido Dagan. 2015a. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.
- Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015b. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, Denver, Colorado.
- Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.
- Bill MacCartney and Christopher D Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 521–528. Association for Computational Linguistics.
- Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 1–8, Dublin, Ireland.

- Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*, pages 48–53, Prague, Czech Republic, June. Association for Computational Linguistics.
- Oren Melamud, Ido Dagan, and Jacob Goldberger. 2015a. Modeling word meaning in context with substitute vectors. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 472–482, Denver, Colorado, May–June. Association for Computational Linguistics.
- Oren Melamud, Omer Levy, and Ido Dagan. 2015b. 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, June. Association for Computational Linguistics.
- Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany, August. Association for Computational Linguistics.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of 2013 International Conference on Learning Representations*.
- George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.
- Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*, pages 809–816.
- Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.
- Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia, June. Association for Computational Linguistics.
- John Alan Robinson. 1965. A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM)*, 12(1):23–41.
- Stephen Roller and Katrin Erk. 2016a. PIC a different word: A simple model for lexical substitution in context. In *Proceedings of the 2016 North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June.
- Stephen Roller and Katrin Erk. 2016b. Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, USA, November. Association for Computational Linguistics.
- Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the 2014 International Conference on Computational Linguistics*, pages 1025–1036, Dublin, Ireland.
- Enrico Santus. 2013. SLQS: An entropy measure. Master’s thesis, University of Pisa.
- Eyal Shnarch. 2008. Lexical entailment and its extraction from wikipedia. Master’s thesis, Bar-Ilan University.

- Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2389–2398, Berlin, Germany, August. Association for Computational Linguistics.
- Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1297–1304. MIT Press.
- Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2006. Semantic taxonomy induction from heterogeneous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 801–808. Association for Computational Linguistics.
- Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013a. Reasoning with neural tensor networks for knowledge base completion. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 926–934. Curran Associates, Inc.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.
- Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Interspeech*, pages 194–197.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.
- György Szarvas, Chris Biemann, and Iryna Gurevych. 2013. Supervised all-words lexical substitution using delexicalized features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1131–1141.
- Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 948–957, Uppsala, Sweden, July. Association for Computational Linguistics.
- Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *The 33rd International Conference on Machine Learning*.
- Peter D Turney and Saif M Mohammad. 2015. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering*, 21(03):437–476.
- Peter Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.
- Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. 2011. Latent vector weighting for word meaning in context. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1012–1022, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.
- Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 81–88.

- Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 2004 International Conference on Computational Linguistics*, pages 1015–1021, Geneva, Switzerland.
- Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of the 2014 International Conference on Computational Linguistics*, pages 2249–2259, Dublin, Ireland.
- Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. In *Proceedings of 2016 International Conference on Learning Representations*.
- Benjamin J. Wilson and Adriaan M. J. Schakel. 2015. Controlled experiments for word embeddings. *ArXiv e-prints*, abs/1510.02675, October.
- Ludwig Wittgenstein. 1953. *Philosophical investigations*. Blackwell Publishers, Oxford, England.
- Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of 2014 International Conference on Learning Representations*.
- Maayan Zhitomirsky-Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 2005 Annual Meeting of the Association for Computational Linguistics*, pages 107–114, Ann Arbor, Michigan.