Relations such as Hypernymy: Identifying and Exploiting Hearst Patterns in Distributional Vectors for Lexical Entailment

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

Katrin Erk  
Department of Linguistics  
The University of Texas at Austin  
katrin.erk@mail.utexas.edu

Abstract

We consider the task of predicting lexical entailment using distributional vectors. We focus experiments on one previous classifier which was shown to only learn to detect prototypicality of a word pair. Analysis shows that the model single-mindedly learns to detect Hearst Patterns, which are well known to be predictive of lexical relations. We present a new model which exploits this Hearst Detector functionality, matching or outperforming prior work on multiple data sets.

1 Introduction

As the field of Natural Language Processing has developed, more ambitious semantic tasks are starting to be addressed, such as Question Answering (QA) and Recognizing Textual Entailment (RTE). These systems often depend on the use of lexical resources like WordNet in order to infer entailment decisions for individual words, but these resources are expensive to develop, and always have limited coverage.

To address these issues, much work has focused on how lexical entailments can be derived automatically using distributional semantics. Some focus mostly on the use of unsupervised techniques, and study measures which emphasize particular word relations (Baroni and Lenci, 2011). Many are based on the Distributional Inclusion Hypothesis, which states that the contexts in which a hypernym appears are a superset of its hyponyms’ contexts (Zhitomirsky-Geffet and Dagan, 2005; Kotlerman et al., 2010). More recently, a great deal of work has pushed toward using supervised methods (Baroni et al., 2012; Roller et al., 2014; Weeds et al., 2014; Levy et al., 2015; Kruszewski et al., 2015), varying mostly by their experimental setup or proposed model.

Yet the literature disagrees about which models are strongest (Weeds et al., 2014; Roller et al., 2014), or even if they work at all (Levy et al., 2015). Indeed, the latter showed that two existing lexical entailment models fail to account for similarity between the antecedent and consequent, and conclude that such models are only learning to predict prototypicality: that is, they predict that cat entails animal because animal is usually entailed, and therefore will also predict that sofa entails animal. If such models cannot learn the relationships between words, then why does one particular model stubbornly persist as such a strong baseline? (Weeds et al., 2014; Kruszewski et al., 2015; Levy et al., 2015)

We examine this question in several phases: first, we propose a better evaluation setup which takes into account multiple trials and more data. We consider a few parameters for which models are particularly sensitive, and show that with these parameters tuned, one prototypicality classifier performs nearly as strong or stronger than other proposed models. We perform an analysis and find that in these settings, this model does not just simply learn to memorize the data: instead, it overwhelmingly learns to detect Hearst patterns. Hearst patterns, like “animals such as cats” and “animals including cats,” have long been used to identify specific lexical relations (Hearst, 1992; Snow et al., 2004).

Armed with this knowledge, we propose a new model which classifies word pairs as lexical entail-
ments using a combination of Hearst pattern detection, overall similarity, and Distributional Inclusion. Our model brings together the strengths of different models in the literature, and matches or outperforms prior work on multiple data sets of hypernymy and lexical entailment.

2 Background

Research on lexical entailment using distributional semantics has now spanned more than a decade, and has been approached using both unsupervised (Weeds et al., 2004; Kotlerman et al., 2010; Lenci and Benotto, 2012; Santus, 2013) and supervised techniques (Baroni et al., 2012; Pu et al., 2014; Roller et al., 2014; Weeds et al., 2014; Kruszewski et al., 2015; Levy et al., 2015; Turney and Mohammad, 2015). Most of the work in unsupervised methods are based on the Distributional Inclusion Hypothesis (Zhitomirsky-Geffet and Dagan, 2005), which states the contexts in which a hypernym appear should be a superset over its hyponyms’ contexts.

This work focuses primarily on the supervised works in the literature, which have varied dramatically in experimental setup, approach, and data set. In examining and reimplementing some of the literature, we hope to bring new clarity and perspective on this difficult task.

Formally, we consider methods which treat lexical entailment as a supervised classification problem, which take as input the distributional vectors for a pair of words, \((H, w)\), and predict on whether the antecedent \(w\) entails the consequent \(H\).

One of the earliest proposed approaches was Concat (Baroni et al., 2012). In this work, the concatenation of the pair \((H, w)\) was used as input to an off-the-shelf SVM classifier. At the time, it was very successful, but later works noted that it had major problems with lexical memorization (Roller et al., 2014; Weeds et al., 2014; Levy et al., 2015). That is, when the training and test sets were carefully constructed to ensure they were completely disjoint, it performed extremely poorly. Nonetheless, Concat is continually used as a strong baseline in more recent work (Kruszewski et al., 2015).

In response to these issues of lexical memorization, alternative models were proposed. Of particular note are the Diff (Fu et al., 2014; Weeds et al., 2014) and Asym classifiers (Roller et al., 2014). The Diff model takes the vector difference, \(H - w\) as input, while the Asym model uses both the vector difference and the squared vector difference as input. Weeds et al. (2014) found that Concat moderately outperformed Diff, while Roller et al. (2014) found that Asym outperformed Concat. Both Diff and Asym can also be seen as a form of supervised Distributional Inclusion Hypothesis, with the vector difference being analogous to the set-inclusion measures of some unsupervised techniques (Roller et al., 2014). All of these works focused exclusively on hypernymy detection, rather than the more general task of lexical entailment.

Recently, other works have begun to analyze Concat and Diff for their ability to go beyond just hypernymy detection. Vylomova et al. (2015) take an extensive look at Diff’s ability to model a wide variety of lexical relations and conclude it is generally robust, and Kruszewski et al. (2015) have success with a neural network model based on the Distributional Inclusion Hypothesis.

On the other hand, Levy et al. (2015) analyze both Concat and Diff in their ability to detect general lexical entailment on several data sets: two consisting of only hypernymy, and three more covering a wide variety of other entailing word relations (c.f. Section 3). They find that both Concat and Diff fail, and analytically show that both are learning to predict how prototypical the consequent \(H\) is, rather than the relationship between the antecedent and the consequent. They propose a new model, Ksim, which addresses their concerns, but lacks any notion of Distributional Inclusion. In particular, they argue in favor of directly including the cosine similarity of \(w\) and \(H\) into models to determine whether they are related in the first place. But ultimately they conclude that distributional semantics may be the wrong tool for the job.

Altogether, the community seems to have reached differing conclusions about the performance of lexical entailment classifiers: some find Diff to be strong and outperform Concat; others find Concat to be the best; and yet others find them both to be ineffectual altogether. Some find Distributional Inclusion to be
important, yet others shy away. What can one possibly discern from this hodgepodge of results?

In the next few sections of this paper, we highlight and illuminate some of the important differences in data, methodology, distributional spaces, and classifiers, allowing us to emphasize the strengths and weaknesses of the Concat, Diff, Asym and Ksim models. We then propose a new classifier in Section 8 which integrates the strengths of all of these models.

3 Data and Resources

As with nearly all modern NLP methods, the most important ingredient is the data. Prior work on lexical entailment relied on a variety of data sets, each constructed in a different manner. We focus on four different data sets, each of which has been used for evaluation in prior work. Two data sets contain only hypernymy relations, and two consider general lexical entailment.

Our first data set is **LEDS**, the Lexical Entailment Data Set, originally created by Baroni et al. (2012). The data set contains 1385 hyponym-hypernym pairs extracted directly from WordNet, forming a set of positive examples. Negative examples were generated by randomly shuffling the original set of 1385 pairs. As such, LEDS only contains examples of hypernymy and random relations.

Another major data set has been **BLESS**, the Evaluation of Semantic Spaces. The data set 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. In this work, we treat hypernymy as positive examples, and other relations as negative. These two data sets form our hypernymy data sets, but we cannot overstate their important differences: LEDS is balanced, while BLESS contains mostly negative examples; Negatives in BLESS include both random pairs and pairs exhibiting other strong semantic relations, while LEDS only contains random pairs. Furthermore, all of the negative examples in LEDS are the same lexical items as the positive items, which has strong implications on the prototypicality argument of Levy et al. (2015).

The next data set we consider is **Medical** (Levy et al., 2014). This data set contains high quality annotations of subject-verb-object entailments extracted from medical texts, and transformed into noun-noun entailments by argument alignments. The data contains 12,600 annotations, but only 945 positive examples encompassing various relations like hypernymy, meronomy, synonymy and contextonymy. This makes it one of the most difficult data sets: it is both domain specific and highly unbalanced.

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

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

3.1 Distributional Vectors

In all experiments, we derive standard count-based distributional vector spaces. We use a corpus composed of the concatenation of Gigaword, Wikipedia, BNC and ukWaC. We preprocess the corpus using Stanford CoreNLP 3.5.2 (Chen and Manning, 2014) for tokenization, lemmatization, POS-tagging and universal dependency parses, and only compute vectors for the 250k most frequent lemmas. We use bag-of-words distributional spaces with different window sizes, as well as a dependency space. For our bag-of-words distributional spaces, we count co-occurrences of the top 20k most frequent content words (JJ, NN, VB, RB) within a fixed window to the left and right of the target. For the dependency vectors, we count the top 1M most frequent dependency attachments as contexts. We use bag-of-words distributional spaces with different window sizes, as well as a dependency space. For our bag-of-words distributional spaces, we count co-occurrences of the top 20k most frequent content words (JJ, NN, VB, RB) within a fixed window to the left and right of the target. For the dependency vectors, we count the top 1M most frequent dependency attachments as contexts. Prepositional dependencies are collapsed e.g. “go to the store” emits the

---

2 A term used in the literature for entailments that occur in some contexts, but do not cleanly fit in other categories; for example, hospital entails doctor.
tuples (go, prep:to+store) and (store, prep:to−1+go). After collecting counts, vectors are transformed using PPMI, SVD reduced to 300 dimensions, and normalized to have unit magnitude.

4 Experimental Setup and Evaluation

We perform two sets of experiments; in Sections 5 through 7, we analyze models from prior work, which will then lead us to a new model and an analysis of that model in Section 8.

In our experiments, we use a variation of 20-fold cross validation which accounts for lexical overlap. We pool all the words from the antecedent (LHS) side of the data into a set and split these into 20 distinct cross-validation groups. If $s$ is the test group for a fold, then the test set for that fold is all pairs $(H, w)$ where $w \in s$. The training set for each fold is then chosen as all remaining pairs which contain no overlap with the test set on either the LHS or the RHS, ensuring zero lexical overlap between training and testing, and that every pair is tested exactly once. This means that all test sets are approximately the same size, but training sizes can vary. The random seed is fixed to ensure all models receive exactly the same train/test data across folds.

This setup differs from those of previous works like Kruszewski et al. (2015) and Levy et al. (2015), who both use single, fixed train/test sets without lexical overlap. We find our setup has several advantages over fixed sets. First, we find there is considerable variance in performance across folds, indicating that multiple trials are necessary. Second, by using a CV-like set up, we are able to train over and evaluate every item in the data set, making much more efficient and representative use of the original data set, while the fixed setups consistently discard roughly half the data as ineligible for either training or test.

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

In all models, we use the default hyperparameters provided by Scikit-Learn (Pedregosa et al., 2011), with the exception of using balanced class weights. Without balanced class weights, many of the models learn degenerate decision functions (e.g. always guess nonentailing).

5 Sensitive Parameters

In this section, we consider two parameters to which we found various models to be exceptionally sensitive: window size of the distributional space, and which off-the-shelf classifier is used. Although we show comparisons for the Concat classifier, we also found results were similar for Diff and Asym as well.

Figure 1 shows the performance for the Concat classifier for varying window sizes of the distributional space. It is well known in the literature that syntactic and small window distributional spaces tend to strongly model word similarity in terms of selectional preferences, while larger windows tend to model more topical similarity (Agirre et al., 2009; Peirsman, 2008; Baroni and Lenci, 2011). Figure 1 shows that performance for all four data sets is best with the purely syntactic space, and degrades substantially with larger windows. Indeed, with a syntactic window, performance improves for every model we consider in this work, giving us numbers higher than those in the literature.

The second main parameter we consider is which off-the-shelf classifier is used with the Concat model. We consider three kinds of Support Vector Machines: a linear kernel, a Radial Basis Kernel, and a Polynomial kernel. We also consider Logistic regression, another linear classifier.
6 Model Comparisons

With these few hyperparameters tuned, we are now ready to make a comparison of the various models proposed in the literature. We compare the Concat, Diff, Asym and Ksim models on all four data sets. All models use the same distributional space and Logistic Regression classifier (except for Ksim). We also include a Cosine baseline which uses a threshold tuned to maximize F1 score on the training set.

Table 1 shows the results for the different classifiers in the different data sets. Our first observation is that the performance varies wildly between the different data sets: the Medical data set has much lower F1 scores than the others, confirming that this is the most difficult data set, partially because of its extreme imbalance, and partially because of its mix of different lexical entailment relations.

We also observe that the ranking of models is inconsistent across the different data sets: while Ksim performs the best on three of the four models, Concat beats it handily on BLESS. Concat outperforms Diff and Asym on BLESS, Medical and TM14, but performs considerably worse than Asym on LEDS. Even comparing to the Cosine baseline can be confusing: Cosine performs nearly as strongly as the supervised models on LEDS and TM14, but dramatically lower on BLESS and Medical. We hope to explain these results by considering what each data set captures and represents, and why each model is more or less equipped for the data.

To begin, we review the findings of Levy et al. (2015), who showed that Concat trained using a linear classifier is only able to capture notions of prototypicality; that is, Concat guesses that (animal, sofa) is an example of hypernymy, because animal looks like other hypernyms. Formally, a linear classifier like Logistic Regression or Linear SVM learns a decision hyperplane \( \hat{p} \). Data points are compared to this plane with the inner product: those above the plane are classified as entails, and those below as nonentails. (Levy et al., 2015) showed one can rewrite this decision hyperplane \( \hat{p} \):

\[
\begin{align*}
\text{Linear}(\langle H, w \rangle) &= \hat{p}^T \langle H, w \rangle \\
&= \langle \hat{H}, \hat{w} \rangle^T \langle H, w \rangle \\
&= \hat{H}^T H + \hat{w}^T w
\end{align*}
\]

Thus, the model only learns to capture the notion of hypernymy through \( \hat{H} \) and \( \hat{w} \), the prototypical vectors. It has absolutely no notion of similarity be-
between $H$ and $w$, and therefore is unable to capture the relationship between the two.

This interpretation of Concat explains why it is especially prone to failure on LEDS: since negative examples are created by randomly shuffling the pairs, the same words appear frequently both as positive and negative examples. As such, Concat is unable to learn what a prototypical hypernym looks like.

This also explains why Asym and Ksim perform so much better: since most negative examples are completely unrelated words, any measure of semantic similarity, like cosine for Ksim and Euclidean distance for Asym, is highly discriminative.

The construction of BLESS also gives insight into why Cosine, Asym and Ksim perform so poorly. The cosine measure of similarity is well-known to emphasize co-hyponyms over every other relation (Baroni and Lenci, 2011; Roller et al., 2014). Since this data set contains a large number of both random pairs and co-hyponym pairs, cosine acts as both a positive signal of hypernymy for the random pairs, and a negative signal for the co-hyponym pairs. Thus, methods like the baseline, Asym and Ksim, which all require Cosine to positively correlate with hypernymy, fail on this data set.

However, Medical and TM14 contain no examples of random pairs: all terms in Medical come from the same domain, meaning general semantic similarity is already reasonably high; and in TM14, all word pairs are in some relationship to each other (like cause and effect), just not necessarily entailing ones. So for these data sets, cosine’s effectiveness is relatively poor.

As expected, the sophisticated Ksim model performs best on three of the data sets: LEDS, Medical and TM14. This is consistent with Levy et al. (2015), who found it to be state of the art. But we are still left with one major mystery: for BLESS, Medical and TM14, why is Concat such a competitive baseline? The analysis of Concat as a prototypicality model suggests that it should always perform poorly. We turn to additional analysis for insight.

7 Analysis

We first perform an analysis of the learned hyperplane $\hat{H}$ to see if it exhibits strong prototypicality behavior. Using the observation from Equation 1, we extract the vocabulary words most similar to the prototype $\hat{H}$ learned by the Concat model. Bold items appear in the data set.

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

Table 2: Most similar words to the prototype $\hat{H}$ learned by the Concat model. Bold items appear in the data set.

Curiously, there are far fewer bold words than we expect if the model were only memorizing the most common words in the data. In LEDS, we actually see some hypernyms of data set items that do not even appear in the data set, and the Medical and TM14 words do not even appear related to the content of the data sets. These lists provide minimal explanation as to Concat’s impressive performance in each task. We propose instead a different view of the hyperplane learned: that of a feature detector.

Recall that distributional vectors are derived from a matrix $M$ containing counts of how often words co-occur with the different syntactic contexts. This cooccurrence matrix is factorized using the Singular Value Decomposition, producing both $W$, the ubiquitous word-embedding matrix, and $C$, the context-embedding matrix (Levy and Goldberg, 2014):

$$M \approx WC^T$$

Since the word and context embeddings implicitly live in the same vector space (Melamud et al., 2015), we can also compare Concat’s hyperplane with the context matrix $C$. Table 3 shows the syntactic contexts with the highest cosine similarity to the $\hat{H}$ prototype for each of the different data sets.

This view of Concat as a feature detector produces a radically different perspective on the classifier’s hyperplane. Most of the features learned all take the form of Hearst patterns (Hearst, 1992; Snow et al., 2004). The most recognizable and common pattern learned is the “such as” pattern, as in “animals such as cats”. These patterns have been well known to be indicative of hypernymy for over two decades.
Other interesting patterns are the “including” pattern (“animals including cats”) and “many” pattern (“many animals”). Taken together, it is remarkable that the model identified such patterns with only the positive and negative examples provided. [Levy et al. (2015)] also reported weak evidence that Concat learns Hearst patterns, but our results are much more dramatic. This is because we compare the prototype to the context embeddings, rather than learning on sparse count features, and because we use a syntactic-distributional space.

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

We hypothesize the TM14 data set contains too many diverse and mutually exclusive forms of lexical entailment, like instrument-goal (e.g. “honey” → “sweetness”). To test this, we retrained the model with only hypernymy as positive examples, and all other relations as negative. We find that “such as” type patterns become top features, but also some interesting data specific features, like “retailer of [clothes]”. Examining the data shows it contains many number of consumer goods, like “beverage” or “clothes”, which partially explains these features.

### 8 Proposed 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 Ksim and Cosine. To this end, we propose a novel model which exploits the Concat classifier, extends its modeling power, and adds two other types of evidence proposed in the literature: overall 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 removed from the data.

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 $\vec{p}$, $\text{proj}_p(x)$, finds the component of $x$ which is in the direction of $\vec{p}$,

$$\text{proj}_p(x) = \left( x^T \frac{\vec{p}}{||\vec{p}||} \right) \vec{p}. $$

Figure 3 gives a geometric illustration of the vector projection. If $x$ forms the hypotenuse of a right triangle, $\text{proj}_p(x)$ 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 the original vector after the projection has been removed:

$$\text{rej}_p(x) = x - \text{proj}_p(x). $$

Using the vector rejection, we take a learned Hearst Pattern detector $\vec{p}$, 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

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

Table 3: Most similar contexts to the prototype $\hat{H}$ learned by the Concat model.
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 \((H', w')\) data must now learn a very different decision plane than \(\hat{p}\), as \(\hat{p}\) is no longer present in any data points. This new classifier must perform strictly worse than the original, otherwise the first classifier would have learned this hyperplane. Nonetheless, it will be able to learn new Hearst patterns which the original classifier was unable to capture. By repeating this process several times, we can find several Hearst pattern detectors, \(\hat{p}_1, \ldots, \hat{p}_n\).

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

\[
F_i((H_i, w_i), \hat{p}_i) = (H_i^T w, H_i^T \hat{p}_i, w_i^T \hat{p}_i, H_i^T \hat{p}_i - w_i^T \hat{p}_i)
\]

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

The union of all the feature vectors \(F_1, \ldots, F_n\) 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 to do well.

The only free hyperparameter of the model is \(n\), the number of iterations of the PCA-like procedure. We found that performance improves substantially on all data sets with at least two iterations, but then falls off slowly after five or more iterations. We chose to use four iterations for all final models.

Table 4 shows the final results of our classifier, compared the Concat and Ksim models of before. Our proposed model improves substantially over Concat in all four data sets, indicating the benefits of combining these the aspects of similarity, feature inclusion and distributional inclusion. The model performs approximately the same same as Ksim on LEDS and TM14, and outperforms it handily on the BLESS and Medical data sets.

In order to evaluate how important each of the various \(F\) features are to the model, we also performed an ablation experiment where the classifier is not given the Similarity (slot 1), Hearst pattern (slots 2 and 3) or the Inclusion features (slot 4). These results are also in Table 4. Without the similarity features, the model performs about the same as the Concat classifier, emphasizing its importance. Without the detector features, the model falls far below Concat, with the exception of LEDS. As mentioned earlier, the construction of LEDS makes Hearst features less salient. Finally, we see that without the Inclusion features, the classifier performs slightly lower, lending further evidence for the Distributional Inclusion Hypothesis. In general, we find Hearst patterns, general similarity, and Distributional Inclusion are all valuable sources of information for detecting hypernymy and lexical entailment.

We also repeat the procedure performed in Sec-
Table 5: Most similar contexts to the Hearst pattern detector for each iteration of the procedure. This model was trained on all data of BLESS. The first and second iterations contain clear Hearst patterns, while the third and fourth are more difficult to discern.

| Iteration 1                     | Iteration 2                              | Iteration 3                              | Iteration 4                              |
|--------------------------------|------------------------------------------|------------------------------------------|------------------------------------------|
| nmod: such as + submarine       | nmod: including + animal                 | amod: free-swimming                       | advcl: crown                            |
| nmod: such as + ship            | nmod: including + snail                  | nmod: including −1 + thing               | advcl: victorious                        |
| nmod: such as + seal            | nmod: including + insect                 | nsubj −1 + scarcer                       | nsubj: eaters                            |
| nmod: such as + plane           | nmod: such as + crustacean               | nsubj −1 + pupate                        | nsubj: kaine                            |
| nmod: such as + rack            | nmod: such as + mollusc                  | nmod: such as + mollusc                  | nmod: at + finale                       |

9 Conclusion

We considered the task of detecting lexical entailment using distributional vectors meaning. We looked closely into the various data sets, distributional spaces and classifiers to discern what are the advantages and disadvantages of the different models. We found that Concat’s performance was particularly compelling, despite previous work showing that it only learns about prototypical hypernyms. An analysis of the Concat classifier in particular showed that, with the right input, it overwhelmingly learns to detect Hearst patterns.

We proposed a new model which exploits and extends this Hearst pattern behavior, while also incorporating useful information from general similarity and Distributional Inclusion. The integration of all three of these signals leads to performance matching our exceeding that of prior work, both on hypernymy detection and general lexical entailment.

Acknowledgments

The authors would like to thank I. Beltagy and Subhashini Venugopalan for their helpful comments and suggestions. This research was supported by the NSF grant IIS 1523637. We acknowledge the Texas Advanced Computing Center for providing grid resources that contributed to these results.

References

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

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

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 Linguists, pages 23–32, Avignon, France.

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

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.

Marti A Hearst. 1992. Automatic acquisition of hypernyms from large text corpora. In Proceedings of the
Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16:359–389.

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

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.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185.

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, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. 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.

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

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

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

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.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems*, pages 1297–1304.

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

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2015. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. *arXiv preprint arXiv:1509.01692 [cs.CL]*.

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.

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.