Distinguishing Antonyms and Synonyms in a Pattern-based Neural Network

Kim Anh Nguyen and **Sabine Schulte im Walde** and **Ngoc Thang Vu**
Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
Pfaffenwaldring 5B, 70569 Stuttgart, Germany
{nguyenkh,schulte,thangvu}@ims.uni-stuttgart.de

Abstract

Distinguishing between antonyms and synonyms is a key task to achieve high performance in NLP systems. While they are notoriously difficult to distinguish by distributional co-occurrence models, pattern-based methods have proven effective to differentiate between the relations. In this paper, we present a novel neural network model *AntSynNET* that exploits lexico-syntactic patterns from syntactic parse trees. In addition to the lexical and syntactic information, we successfully integrate the distance between the related words along the syntactic path as a new pattern feature. The results from classification experiments show that AntSynNET improves the performance over prior pattern-based methods.

1 Introduction

Antonymy and synonymy represent lexical semantic relations that are central to the organization of the mental lexicon (Miller and Fellbaum, 1991). While antonymy is defined as the oppositeness between words, synonymy refers to words that are similar in meaning (Deese, 1965; Lyons, 1977). From a computational point of view, distinguishing between antonymy and synonymy is important for NLP applications such as Machine Translation and Textual Entailment, which go beyond a general notion of semantic relatedness and require to identify specific semantic relations. However, due to interchangeable substitution, antonyms and synonyms often occur in similar contexts, which makes it challenging to automatically distinguish between them.

Two families of approaches to differentiate between antonyms and synonyms are predominent in NLP. Both make use of distributional vector representations, relying on the *distributional hypothesis* (Harris, 1954; Firth, 1957), that words with similar distributions have related meanings: co-occurrence models and pattern-based models. These distributional semantic models (DSMs) offer a means to represent meaning vectors of words or word pairs, and to determine their semantic relatedness (Turney and Pantel, 2010).

In *co-occurrence models*, each word is represented by a weighted feature vector, where features typically correspond to words that co-occur in particular contexts. When using word embeddings, these models rely on neural methods to represent words as low-dimensional vectors. To create the word embeddings, the models either make use of neural-based techniques, such as the skip-gram model (Mikolov et al., 2013), or use matrix factorization (Pennington et al., 2014) that builds word embeddings by factorizing word-context co-occurrence matrices. In comparison to standard co-occurrence vector representations, word embeddings address the problematic sparsity of word vectors and have achieved impressive results in many NLP tasks such as word similarity (e.g., Pennington et al. (2014)), relation classification (e.g., Vu et al. (2016)), and antonym-synonym distinction (e.g., Nguyen et al. (2016)).

In *pattern-based models*, vector representations make use of lexico-syntactic surface patterns to distinguish between the relations of word pairs. For example, Justeson and Katz (1991) suggested that adjectival opposites co-occur with each other in specific linear sequences, such as `between X and Y`. Hearst (1992) determined surface patterns, e.g., `X such as Y`, to identify nominal hypernyms. Lin et al. (2003) proposed two textual patterns indicating semantic incompatibility, `from X to Y` and `either X or Y`, to distinguish opposites from semantically similar words. Roth and Schulte im Walde (2014) proposed a method that combined patterns with discourse markers for classifying paradigmatic relations including antonymy, synonymy, and hypernymy. Recently, Schwartz et al. (2015) used two prominent patterns from Lin et al. (2003) to learn word embeddings that distinguished antonyms from similar words in determining degrees of similarity and word analogy.

In this paper, we present a novel pattern-based neural method *AntSynNET* to distinguish antonyms from synonyms. We hypothesize that antonymous word pairs co-occur with each other in lexico-syntactic patterns within a sentence more often than would be expected by synonymous pairs. This hypothesis is inspired by corpus-based studies on antonymy and synonymy. Among others, Charles and Miller (1989) suggested that adjectival opposites co-occur in patterns; Fellbaum (1995) stated that nominal and verbal opposites co-occur in the same sentence significantly more often than chance; Lin et al. (2003) argued that if two words appear in clear antonym patterns, they are unlikely to represent synonymous pair.

We start out by inducing patterns between `X` and `Y` from a large-scale web corpus, where `X` and `Y` represent two words of an antonym or synonym word pair, and the pattern is derived from the simple paths between `X` and `Y` in a syntactic parse tree. Each node in the simple path combines lexical and syntactic information; in addition, we suggest a novel feature for the patterns, i.e., the distance between the two words along the syntactic path. All pattern features are fed into a recurrent neural network with long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997), which encode the patterns as vector representations. Afterwards, the vector representations of the patterns are used in a classifier to distinguish between antonyms and synonyms. The results from experiments show that AntSynNET improves the performance over prior pattern-based methods. Furthermore, the implementation of our models is made publicly available[1].

The remainder of this paper is organized as follows: In Section 2, we present previous work distinguishing antonyms and synonyms. Section 3 describes our proposed AntSynNET model. We present the induction of the patterns (Section 3.1), describe the recurrent neural network with long short-term memory units which is used to encode patterns within a vector representation (Section 3.2), and describe two models to classify antonyms and synonyms: the pure pattern-based model (Section 3.3.1) and the combined model (Section 3.3.2). After introducing two baselines in Section 4, we describe our dataset, experimental settings, results of our methods, the effects of the newly proposed distance feature, and the effects of the various types of word embeddings. Section 6 concludes the paper.

2 Related Work

Pattern-based methods: Regarding the task of antonym-synonym distinction, there exist a variety of approaches which rely on patterns. Lin et al. (2003) used bilingual dependency triples and patterns to extract distributionally similar words. They relied on clear antonym patterns such as `from X to Y` and `either X or Y` in a post-processing step to distinguish antonyms from synonyms. The main idea is that if two words `X` and `Y` appear in one of these patterns, they are unlikely to represent synonymous pair. Schulte im Walde and Köper (2013) proposed a method to distinguish between the paradigmatic relations antonymy, synonymy and hypernymy in German, based on automatically acquired word patterns. Roth and Schulte im Walde (2014) combined general lexico-syntactic patterns with discourse markers as indicators for the same relations, both for German and for English. They assumed that if two phrases frequently co-occur with a specific discourse marker, then the discourse relation expressed by the corresponding marker should also indicate the relation between the words in the affected phrases. By using the raw corpus and a fixed list of discourse markers, the model can easily be extended to other languages. More recently, Schwartz et al. (2015) presented a symmetric pattern-based model for word vector representation in which antonyms are assigned to dissimilar vector representations. Differently to the previous pattern-based methods which used the standard distribution of patterns, Schwartz et al. used patterns to learn word embeddings.

Vector representation methods: Yih et al. (2012) introduced a new vector representation where antonyms lie on opposite sides of a sphere. They derived this representation with the incorporation of a thesaurus and latent semantic anal-

[1] `https://github.com/nguyenkh/AntSynNET`

ysis, by assigning signs to the entries in the co-occurrence matrix on which latent semantic analysis operates, such that synonyms would tend to have positive cosine similarities, and antonyms would tend to have negative cosine similarities. Scheible et al. (2013) showed that the distributional difference between antonyms and synonyms can be identified via a simple word space model by using appropriate features. Instead of taking into account all words in a window of a certain size for feature extraction, the authors experimented with only words of a certain part-of-speech, and restricted distributions. Santus et al. (2014) proposed a different method to distinguish antonyms from synonyms by identifying the most salient dimensions of meaning in vector representations and reporting a new average-precision-based distributional measure and an entropy-based measure. Ono et al. (2015) trained supervised word embeddings for the task of identifying antonymy. They proposed two models to learn word embeddings: the first model relied on thesaurus information; the second model made use of distributional information and thesaurus information. More recently, Nguyen et al. (2016) proposed two methods to distinguish antonyms from synonyms: in the first method, the authors improved the quality of weighted feature vectors by strengthening those features that are most salient in the vectors, and by putting less emphasis on those that are of minor importance when distinguishing degrees of similarity between words. In the second method, the lexical contrast information was integrated into the skip-gram model (Mikolov et al., 2013) to learn word embeddings. This model successfully predicted degrees of similarity and identified antonyms and synonyms.

3 AntSynNET: LSTM-based Antonym-Synonym Distinction

In this section, we describe the AntSynNET model, using a pattern-based LSTM for distinguishing antonyms from synonyms. We first present the induction of patterns from a parsed corpus (Section 3.1). Section 3.2 then describes how we utilize the recurrent neural network with long short-term memory units to encode the patterns as vector representation. Finally, we present the AntSynNET model and two approaches to classify antonyms and synonyms (Section 3.3).

3.1 Induction of Patterns

Corpus-based studies on antonymy have suggested that opposites co-occur with each other within a sentence significantly more often than would be expected by chance. Our method thus makes use of patterns as the main indicators of word pair co-occurrence, to enforce a distinction between antonyms and synonyms. Figure 1 shows a syntactic parse tree of the sentence *"My old village has been provided with the new services"*. Following the characterizations of a tree in graph theory, any two nodes (vertices) of a tree are connected by a simple path (or one unique path). The simple path is the shortest path between any two nodes in a tree and does not contain repeated nodes. In the example, the lexico-syntactic tree pattern of the antonymous pair *old–new* is determined by finding the simple path (in red) from the lemma `old` to the lemma `new`. It focuses on the most relevant information and ignores irrelevant information which does not appear in the simple path (i.e., *has, been*). The example pattern between `X = old` and `Y = new` in Figure 1 is represented as follows: `X/JJ/amod/2 -- village/NN/nsubj/1 -- provide/VBN/ROOT/0 -- with/IN/prep/1 -- service/NNS/pobj/2 -- Y/JJ/amod/3`.

Node Representation: The path patterns make use of four features to represent each node in the syntax tree: lemma, part-of-speech (POS) tag, dependency label and distance label. The lemma feature captures the lexical information of words in the sentence, while the POS and dependency features capture the morpho-syntactic information of the sentence. The distance label measures the path distance between the target word nodes in the syntactic tree. Each step between a parent and a child node represents a distance of 1; and the ancestor nodes of the remaining nodes in the path are represented by a distance of 0. For example, the node `provided` is an ancestor node of the simple path from `old` to `new`. The distances from the node `provided` to the nodes `village` and `old` are 1 and 2, respectively.

The vector representation of each node concatenates the four-feature vectors as follows:

$$\vec{v}_{node} = [\vec{v}_{lemma} \oplus \vec{v}_{pos} \oplus \vec{v}_{dep} \oplus \vec{v}_{dist}]$$

where $\vec{v}_{lemma}, \vec{v}_{pos}, \vec{v}_{dep}, \vec{v}_{dist}$ represent the embeddings of the lemma, POS tag, dependency label

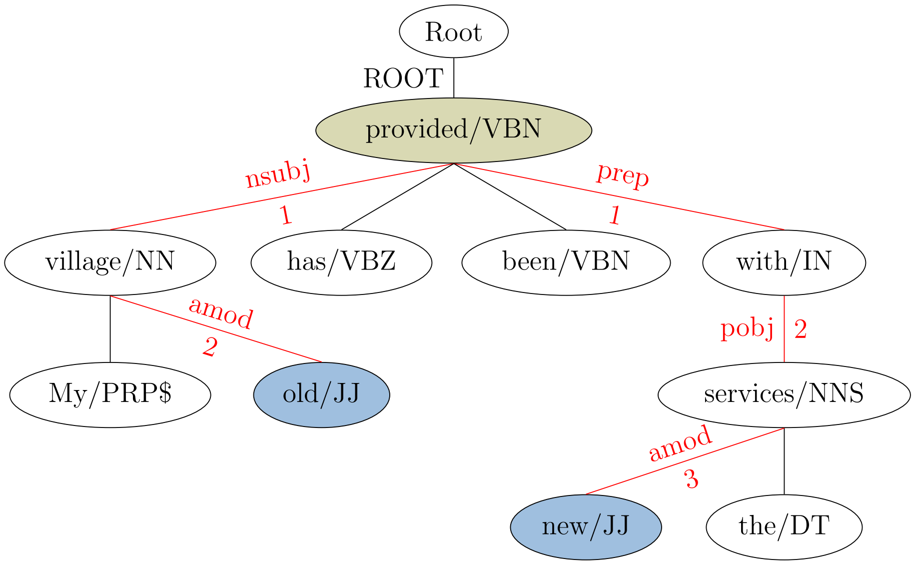

Figure 1: Illustration of the syntactic tree for the sentence *"My old village has been provided with the new services"*. Red lines indicate the path from the word `old` to the word `new`.

and distance label, respectively; and the $\oplus$ denotes the concatenation operation.

Pattern Representation: For a pattern p which is constructed by the sequence of nodes $n_1, n_2, ..., n_k$, the pattern representation of p is a sequence of vectors: $p = [\vec{n}_1, \vec{n}_2, ..., \vec{n}_k]$. The pattern vector $\vec{v}_p$ is then encoded by applying a recurrent neural network.

3.2 Recurrent Neural Network with Long Short-Term Memory Units

A recurrent neural network (RNN) is suitable for modeling sequential data by a vector representation. In our methods, we use a long short-term memory (LSTM) network, a variant of a recurrent neural network to encode patterns, for the following reasons. Given a sequence of words $p = [n_1, n_2, ..., n_k]$ as input data, an RNN processes each word n_t at a time, and returns a vector of state h_k for the complete input sequence. For each time step t, the RNN updates an internal memory state h_t which depends on the current input n_t and the previous state h_{t-1}. Yet, if the sequential input is a long-term dependency, an RNN faces the problem of gradient vanishing or exploding, leading to difficulties in training the model.

LSTM units address these problems. The underlying idea of an LSTM is to use an adaptive gating mechanism to decide on the degree that LSTM units keep the previous state and memorize the extracted features of the current input. More specifically, an LSTM comprises four components: an input gate i_t, a forget gate f_t, an output gate o_t, and a memory cell c_t. The state of an LSTM at each time step t is formalized as follows:

$$\begin{aligned}
i_t &= \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \\
f_t &= \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \\
o_t &= \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o) \\
g_t &= \tanh(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c) \\
c_t &= i_t \otimes g_t + f_t \otimes c_{t-1}
\end{aligned}$$

W refers to a matrix of weights that projects information between two layers; b is a layer-specific vector of bias terms; σ denotes the sigmoid function. The output of an LSTM at a time step t is computed as follows:

$$h_t = o_t \otimes \tanh(c_t)$$

where $\otimes$ denotes element-wise multiplication. In our methods, we rely on the last state h_k to represent the vector $\vec{v}_p$ of a pattern $p = [\vec{n}_1, \vec{n}_2, ..., \vec{n}_k]$.

3.3 The Proposed AntSynNET Model

In this section, we present two models to distinguish antonyms from synonyms. The first model makes use of patterns to classify antonyms and synonyms, by using an LSTM to encode patterns as vector representations and then feeding those vectors to a logistic regression layer (Section 3.3.1). The second model creates combined vector representations of word pairs, which concatenate the vectors of the words and the patterns (Section 3.3.2).

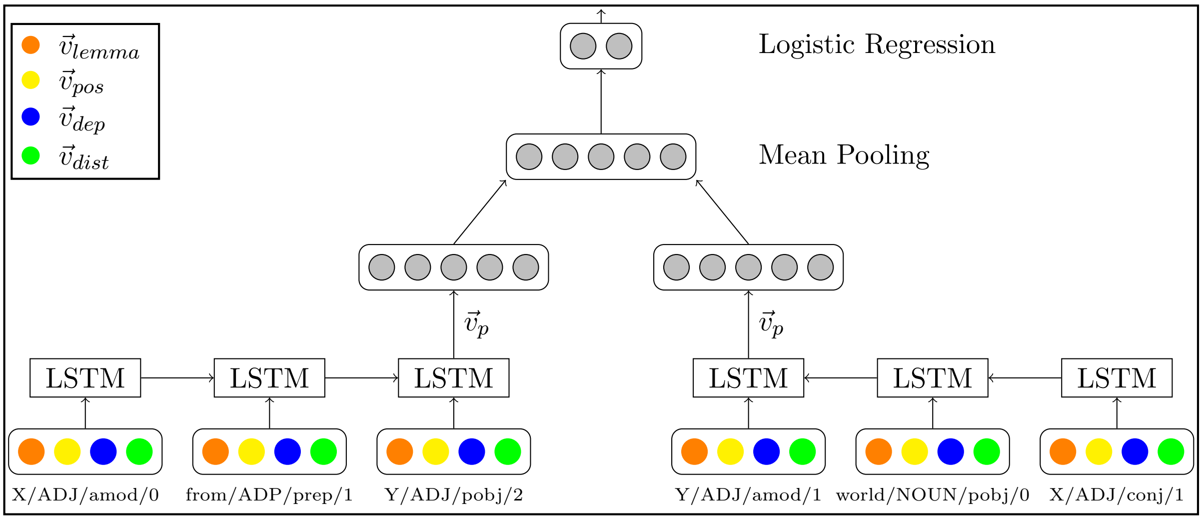

Figure 2: Illustration of the *AntSynNET* model. Each word pair is represented by several patterns, and each pattern represents a path in the graph of the syntactic tree. Patterns consist of several nodes where each node is represented by a vector with four features: lemma, POS, dependency label, and distance label. The mean pooling of the pattern vectors is the vector representation of each word pair, which is then fed to the logistic regression layer to classify antonyms and synonyms.

3.3.1 Pattern-based AntSynNET

In this model, we make use of a recurrent neural network with LSTM units to encode patterns containing a sequence of nodes. Figure 2 illustrates the AntSynNET model. Given a word pair (x, y), we induce patterns for (x, y) from a corpus, where each pattern represents a path from x to y (cf. Section 3.1). We then feed each pattern p of the word pair (x, y) into an LSTM to obtain $\vec{v}_p$, the vector representation of the pattern p (cf. Section 3.2). For each word pair (x, y), the vector representation of (x, y) is computed as follows:

$$\vec{v}_{xy} = \frac{\sum_{p \in P(x,y)} \vec{v}_p \cdot c_p}{\sum_{p \in P(x,y)} c_p} \quad (1)$$

$\vec{v}_{xy}$ refers to the vector of the word pair (x, y); $P(x, y)$ is the set of patterns corresponding to the pair (x, y); c_p is the frequency of the pattern p. The vector $\vec{v}_{xy}$ is then fed into a logistic regression layer whose target is the class label associated with the pair (x, y). Finally, the pair (x, y) is predicted as positive (i.e., antonymous) word pair if the probability of the prediction for $\vec{v}_{xy}$ is larger than 0.5.

3.3.2 Combined AntSynNET

Inspired by the supervised distributional concatenation method in Baroni et al. (2012) and the integrated path-based and distributional method for hypernymy detection in Shwartz et al. (2016), we take into account the patterns and distribution of target pairs to create their combined vector representations. Given a word pair (x, y), the combined vector representation of the pair (x, y) is determined by using both the co-occurrence distribution of the words and the syntactic path patterns:

$$\vec{v}_{comb(x,y)} = [\vec{v}_x \oplus \vec{v}_{xy} \oplus \vec{v}_y] \quad (2)$$

$\vec{v}_{comb(x,y)}$ refers to the combined vector of the word pair (x, y); $\vec{v}_x$ and $\vec{v}_y$ are the vectors of word x and word y, respectively; $\vec{v}_{xy}$ is the vector of the pattern that corresponds to the pair (x, y), cf. Section 3.3.1. Similar to the pattern-based model, the combined vector $\vec{v}_{comb(x,y)}$ is fed into the logistic regression layer to classify antonyms and synonyms.

4 Baseline Models

To compare AntSynNET with baseline models for pattern-based classification of antonyms and synonyms, we introduce two pattern-based baseline methods: the distributional method (Section 4.1), and the distributed method (Section 4.2).

4.1 Distributional Baseline

As a first baseline, we apply the approach by Roth and Schulte im Walde (2014), henceforth R&SiW. They used a vector space model to represent pairs of words by a combination of standard lexico-syntactic patterns and discourse markers. In addition to the patterns, the discourse markers added information to express discourse relations, which in turn may indicate the specific semantic relation between the two words in a word pair. For example, contrast relations might indicate antonymy, whereas elaborations may indicate synonymy or hyponymy.

Michael Roth, the first author of R&SiW, kindly computed the relation classification results of the pattern–discourse model for our test sets. The weights between marker-based and pattern-based models were tuned on the validation sets; other hyperparameters were set exactly as described by the R&SiW method.

4.2 Distributed Baseline

The *SP* method proposed by Schwartz et al. (2015) uses symmetric patterns for generating word embeddings. In this work, the authors applied an unsupervised algorithm for the automatic extraction of symmetric patterns from plain text. The symmetric patterns were defined as a sequence of 3-5 tokens consisting of exactly two wildcards and 1-3 words. The patterns were filtered based on their frequencies, such that the resulting pattern set contained 11 patterns. For generating word embeddings, a matrix of co-occurrence counts between patterns and words in the vocabulary was computed, using positive point-wise mutual information. The sparsity problem of vector representations was addressed by smoothing. For antonym representation, the authors relied on two patterns suggested by Lin et al. (2003) to construct word embeddings containing an antonym parameter that can be turned on in order to represent antonyms as dissimilar, and that can be turned off to represent antonyms as similar.

To apply the SP method to our data, we make use of the pre-trained SP embeddings[2] with 500 dimensions[3]. We calculate the cosine similarity of word pairs and then use a Support Vector Machine with Radial Basis Function kernel to classify antonyms and synonyms.

[2] http://homes.cs.washington.edu/~roysch/papers/sp_embeddings/sp_embeddings.html

[3] The 500-dimensional embeddings outperformed the 300-dimensional embeddings for our data.

5 Experiments

5.1 Dataset

For training the models, neural networks require a large amount of training data. We use the existing large-scale antonym and synonym pairs previously used by Nguyen et al. (2016). Originally, the data pairs were collected from WordNet (Miller, 1995) and Wordnik[4].

In order to induce patterns for the word pairs in the dataset, we identify the sentences in the corpus that contain the word pair. Thereafter, we extract all patterns for the word pair. We filter out all patterns which occur less than five times; and we only take into account word pairs that contain at least five patterns for training, validating and testing. For the proportion of positive and negative pairs, we keep a ratio of 1:1 positive (antonym) to negative (synonym) pairs in the dataset. In order to create the sets of training, testing and validation data, we perform random splitting with 70% train, 25% test, and 5% validation sets. The final dataset contains the number of word pairs according to word classes described in Table 1. Moreover, Table 2 shows the average number of patterns for each word pair in our dataset.

Word Class	Train	Test	Validation	Total
Adjective	5562	1986	398	7946
Verb	2534	908	182	3624
Noun	2836	1020	206	4062

Table 1: Our dataset.

Word Class	Train	Test	Validation
Adjective	135	131	141
Verb	364	332	396
Noun	110	132	105

Table 2: Average number of patterns per word pair across word classes.

5.2 Experimental Settings

We use the English Wikipedia dump[5] from June 2016 as the corpus resource for our methods and baselines. For parsing the corpus, we rely on spaCy[6]. For the lemma embeddings, we rely on the word embeddings of the dLCE

[4] http://www.wordnik.com

[5] https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2

[6] https://spacy.io

Model	Adjective			Verb			Noun		
	P	**R**	$\mathbf{F_1}$	**P**	**R**	$\mathbf{F_1}$	**P**	**R**	$\mathbf{F_1}$
SP baseline	0.730	0.706	0.718	0.560	0.609	0.584	0.625	0.393	0.482
R&SiW baseline	0.717	0.717	0.717	**0.789**	0.787	**0.788**	**0.833**	0.831	0.832
Pattern-based AntSynNET	**0.764**	0.788	0.776*	0.741	**0.833**	0.784	0.804	0.851	0.827
Combined AntSynNET	0.763	**0.807**	**0.784***	0.743	0.815	0.777	0.816	**0.898**	**0.855****

Table 3: Performance of the AntSynNET models in comparison to the baseline models.

Feature	Model	Adjective			Verb			Noun		
		P	**R**	$\mathbf{F_1}$	**P**	**R**	$\mathbf{F_1}$	**P**	**R**	$\mathbf{F_1}$
Direction	Pattern-based	0.752	0.755	0.753	0.734	0.819	0.774	0.800	0.825	0.813
	Combined	0.754	0.784	0.769	0.739	0.793	0.765	**0.829**	0.810	0.819
Distance	Pattern-based	**0.764**	0.788	0.776	0.741	**0.833**	**0.784**	0.804	0.851	0.827
	Combined	0.763	**0.807**	**0.784****	**0.743**	0.815	0.777	0.816	**0.898**	**0.855****

Table 4: Comparing the novel distance feature with Schwarz et al.'s direction feature, across word classes.

model[7] (Nguyen et al., 2016) which is the state-of-the-art vector representation for distinguishing antonyms from synonyms. We re-implemented this cutting-edge model on Wikipedia with 100 dimensions, and then make use of the dLCE word embeddings for initialization the lemma embeddings. The embeddings of POS tags, dependency labels, distance labels, and out-of-vocabulary lemmas are initialized randomly. The number of dimensions is set to 10 for the embeddings of POS tags, dependency labels and distance labels. We use the validation sets to tune the number of dimensions for these labels. For optimization, we rely on the cross-entropy loss function and Stochastic Gradient Descent with the Adadelta update rule (Zeiler, 2012). For training, we use the `Theano` framework (Theano Development Team, 2016). Regularization is applied by a dropout of 0.5 on each of component's embeddings (dropout rate is tuned on the validation set). We train the models with 40 epochs and update all embeddings during training.

5.3 Overall Results

Table 3 shows the significant[8] performance of our models in comparison to the baselines. Concerning adjectives, the two proposed models significantly outperform the two baselines: The performance of the baselines is around .72 for F_1, and the corresponding results for the combined AntSynNET model achieve an improvement of $>$.06. Regarding nouns, the improvement of the new methods is just .02 F_1 in comparison to the R&SiW baseline, but we achieve a much better performance in comparison to the SP baseline, an increase of .37 F_1. Regarding verbs, we do not outperform the more advanced R&SiW baseline in terms of the F_1 score, but we obtain higher recall scores. In comparison to the SP baseline, our models still show a clear F_1 improvement.

Overall, our proposed models achieve comparatively high recall scores compared to the two baselines. This strengthens our hypothesis that there is a higher possibility for the co-occurrence of antonymous pairs in patterns over synonymous pairs within a sentence. Because, when the proposed models obtain high recall scores, the models are able to retrieve most relevant information (antonymous pairs) corresponding to the patterns. Regarding the low precision in the two proposed models, we sampled randomly 5 pairs in each population: true positive, true negative, false positive, false negative. We then compared the overlap of patterns for the true predictions (true positive pairs and true negative pairs) and the false predictions (false positive pairs and false negative pairs). We found out that there is no overlap between patterns of true predictions; and the number overlap between patterns of false predictions is 2, 2, and 4 patterns for noun, adjective, and verb classes, respectively. This shows that the low precision of our models stems from the patterns which represent both antonymous and synonymous pairs.

5.4 Effect of the Distance Feature

In our models, the novel distance feature is successfully integrated along the syntactic path to represent lexico-syntactic patterns. The intu-

[7] `https://github.com/nguyenkh/AntSynDistinction`

[8] t-test, $*p < 0.05$, $**p < 0.1$

Model	Word Embeddings	Adjective			Verb			Noun		
		P	R	F_1	P	R	F_1	P	R	F_1
Pattern-based Model	GloVe	0.763	0.770	0.767	0.705	0.852	0.772	0.789	0.849	0.818
	dLCE	0.764	0.788	**0.776**	0.741	0.833	**0.784**	0.804	0.851	**0.827**
Combined Model	Glove	0.750	0.798	0.773	0.717	0.826	0.768	0.807	0.827	0.817
	dLCE	0.763	0.807	**0.784**	0.743	0.815	**0.777**	0.816	0.898	**0.855**

Table 5: Comparing pre-trained GloVe and dLCE word embeddings.

ition behind the distance feature exploits properties of trees in graph theory, which show that there exist differences in the degree of relationship between the parent node and the child nodes ($distance = 1$) and in the degree of relationship between the ancestor node and the descendant nodes ($distance > 1$). Hence, we use the distance feature to effectively capture these relationships.

In order to evaluate the effect of our novel distance feature, we compare the distance feature to the direction feature proposed by Shwartz et al. (2016). In their approach, the authors combined lemma, POS, dependency, and direction features for the task of hypernym detection. The direction feature represented the direction of the dependency label between two nodes in a path from `X` to `Y`.

For evaluation, we make use of the same information regarding dataset and patterns as in Section 5.3, and then replace the distance feature by the direction feature. The results are shown in Table 4. The distance feature enhances the performance of our proposed models more effectively than the direction feature does, across all word classes.

5.5 Effect of Word Embeddings

Our methods rely on the word embeddings of the dLCE model, state-of-the-art word embeddings for antonym-synonym distinction. Yet, the word embeddings of the dLCE model, i.e., supervised word embeddings, represent information collected from lexical resources. In order to evaluate the effect of these word embeddings on the performance of our models, we replace them by the pre-trained GloVe word embeddings[9] with 100 dimensions, and compare the effects of the GloVe word embeddings and the dLCE word embeddings on the performance of the two proposed models.

Table 5 illustrates the performance of our two models on all word classes. The table shows that the dLCE word embeddings are better than the pre-trained GloVe word embeddings, by around .01 F_1 for the pattern-based AntSynNET model and the combined AntSynNET model regarding adjective and verb pairs. Regarding noun pairs, the improvements of the dLCE word embeddings over pre-trained GloVe word embeddings achieve around .01 and .04 F_1 for the pattern-based model and the combined model, respectively.

[9] `http://www-nlp.stanford.edu/projects/glove/`

6 Conclusion

In this paper, we presented a novel pattern-based neural method *AntSynNET* to distinguish antonyms from synonyms. We hypothesized that antonymous word pairs co-occur with each other in lexico-syntactic patterns within a sentence more often than synonymous word pairs.

The patterns were derived from the simple paths between semantically related words in a syntactic parse tree. In addition to lexical and syntactic information, we suggested a novel path distance feature. The AntSynNET model consists of two approaches to classify antonyms and synonyms. In the first approach, we used a recurrent neural network with long short-term memory units to encode the patterns as vector representations; in the second approach, we made use of the distribution and encoded patterns of the target pairs to generate combined vector representations. The resulting vectors of patterns in both approaches were fed into the logistic regression layer for classification.

Our proposed models significantly outperformed two baselines relying on previous work, mainly in terms of recall. Moreover, we demonstrated that the distance feature outperformed a previously suggested direction feature, and that our embeddings outperformed the state-of-the-art GloVe embeddings. Last but not least, our two proposed models only rely on corpus data, such that the models are easily applicable to other languages and relations.

Acknowledgements

We would like to thank Michael Roth for helping us to compute the results of the R&SiW model on our dataset.

The research was supported by the Ministry of Education and Training of the Socialist Republic of Vietnam (Scholarship 977/QD-BGDDT; Kim-Anh Nguyen), the DFG Collaborative Research Centre SFB 732 (Kim-Anh Nguyen, Ngoc Thang Vu), and the DFG Heisenberg Fellowship SCHU-2580/1 (Sabine Schulte im Walde).

References

[Baroni et al.2012] Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 23–32, Avignon, France.

[Charles and Miller1989] Walter G. Charles and George A. Miller. 1989. Contexts of antonymous adjectives. *Applied Psychology*, 10:357–375.

[Deese1965] James Deese. 1965. *The Structure of Associations in Language and Thought*. The John Hopkins Press, Baltimore, MD.

[Fellbaum1995] Christiane Fellbaum. 1995. Co-occurrence and antonymy. *International Journal of Lexicography*, 8:281–303.

[Firth1957] John R. Firth. 1957. *Papers in Linguistics 1934-51*. Longmans, London, UK.

[Harris1954] Zellig S. Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

[Hearst1992] Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *In Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pages 539–545, Nantes, France.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[Justeson and Katz1991] John S. Justeson and Slava M. Katz. 1991. Co-occurrences of antonymous adjectives and their contexts. *Computational Linguistics*, 17:1–19.

[Lin et al.2003] Dekang Lin, Shaojun Zhao, Lijuan Qin, and Ming Zhou. 2003. Identifying synonyms among distributionally similar words. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1492–1493, Acapulco, Mexico.

[Lyons1977] John Lyons. 1977. *Semantics*, volume 1. Cambridge University Press.

[Mikolov et al.2013] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 746–751, Atlanta, Georgia.

[Miller and Fellbaum1991] George A. Miller and Christiane Fellbaum. 1991. Semantic networks of English. *Cognition*, 41:197–229.

[Miller1995] George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

[Nguyen et al.2016] Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 454–459, Berlin, Germany.

[Ono et al.2015] Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 984–989, Denver, Colorado.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

[Roth and Schulte im Walde2014] Michael Roth and Sabine Schulte im Walde. 2014. Combining word patterns and discourse markers for paradigmatic relation classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 524–530, Baltimore, MD.

[Santus et al.2014] Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 38–42, Gothenburg, Sweden.

[Scheible et al.2013] Silke Scheible, Sabine Schulte im Walde, and Sylvia Springorum. 2013. Uncovering distributional differences between synonyms and antonyms in a word space model. In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 489–497, Nagoya, Japan.

[Schulte im Walde and Köper2013] Sabine Schulte im Walde and Maximilian Köper. 2013. Pattern-based distinction of paradigmatic relations for german nouns, verbs, adjectives. In *Proceedings of the 25th International Conference of the German Society for Computational Linguistics and Language Technology (GSCL)*, pages 189–198, Darmstadt, Germany.

[Schwartz et al.2015] Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of the 19th Conference on Computational Language Learning (CoNLL)*, pages 258–267, Beijing, China.

[Shwartz et al.2016] 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 (ACL)*, pages 2389–2398, Berlin, Germany.

[Theano Development Team2016] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

[Turney and Pantel2010] Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

[Vu et al.2016] Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 534–539.

[Yih et al.2012] Wen-tau Yih, Geoffrey Zweig, and John C. Platt. 2012. Polarity inducing latent semantic analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP)*, pages 1212–1222, Jeju Island, Korea.

[Zeiler2012] Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.