Antonym-Synonym Classification Based on New Sub-Space Embeddings

Muhammad Asif Ali,[1] **Yifang Sun,**[1] **Xiaoling Zhou,**[1] **Wei Wang,**[1,2] **Xiang Zhao**[3]
[1]School of Computer Science and Engineering, UNSW, Australia
[2]College of Computer Science and Technology, DGUT, China
[3]Key Laboratory of Science and Technology on Information System Engineering, NUDT, China
muhammadasif.ali@unsw.edu.au, {yifangs, xiaolingz, weiw}@cse.unsw.edu.au, xiangzhao@nudt.edu.cn

Abstract

Distinguishing antonyms from synonyms is a key challenge for many NLP applications focused on the lexical-semantic relation extraction. Existing solutions relying on large-scale corpora yield low performance because of huge contextual overlap of antonym and synonym pairs. We propose a novel approach entirely based on pre-trained embeddings. We hypothesize that the pre-trained embeddings comprehend a blend of lexical-semantic information and we may distill the task-specific information using Distiller, a model proposed in this paper. Later, a classifier is trained based on features constructed from the distilled sub-spaces along with some word level features to distinguish antonyms from synonyms. Experimental results show that the proposed model outperforms existing research on antonym synonym distinction in both speed and performance.

1 Introduction

Distinguishing between antonymy and synonymy is one of the crucial problems for NLP applications, especially those focused on lexical-semantic relation extraction, such as sentiment analysis, semantic relatedness, opinion mining and machine translation. We define synonyms as semantically similar words (e.g., `disperse` and `scatter`) and antonyms as highly contrasting words (e.g., `disperse` and `garner`) (Ono, Miwa, and Sasaki 2015). Existing manually curated lexical resources (e.g., WordNet (Fellbaum 1998)) are unable to address this problem, due to the limited coverage. This calls the need for a machine learning model to classify a given pair of words as either synonyms or antonyms.

Traditional research on antonym/synonym distinction makes use of word embeddings to capture the semantic relatedness among antonym/synonym pairs based on co-occurrence statistics (Scheible, Schulte im Walde, and Springorum 2013; Ono, Miwa, and Sasaki 2015; Nguyen, Schulte im Walde, and Vu 2016; Vulic 2018). However, the embedding models have a tendency to mix different lexico-semantic relations, so the performance cannot be guaranteed when they are applied on specific lexical-semantic analysis tasks (Glavas and Vulic 2018). This situation worsens in the case of antonym and synonym distinction, because these words can be used interchangeably and are considered indistinguishable (Mohammad et al. 2013). This is verified in this paper by the poor performance of a baseline classifier, i.e., Direct, that is purely designed based on pre-trained word embeddings.

Recently, pattern based approaches have gained considerable research attention for lexical-semantic relation extraction (Roth and Schulte im Walde 2014; Schwartz, Reichart, and Rappoport 2015). Nguyen, Schulte im Walde, and Vu (2017) formulated special lexico-syntactic pattern features that concatenate the lemma, POS-tag and dependency label along the dependency path. Their model achieved the state-of-the-art result on antonym/synonym distinction task. Pattern based methods yield a low recall owing to the lexical variations and the sparsity of lexico-syntactic patterns, which limits the information sharing among semantically similar patterns. Existing attempts to resolve the sparsity via generalization cause the resultant patterns to be highly overlapping across different classes, which has a detrimental effect on the classification accuracy. Moreover, they have to use a large-scale text corpus to combat sparsity, which drastically increases the computational overhead.

In this paper, we propose a novel two-phase approach to address the above-mentioned challenges for antonym/synonym distinction by eliminating the need for a large text corpus. Firstly, we use a new model named: Distiller, which is a set of non-linear encoders, to distill task-specific information from *pre-trained word embeddings* to dense sub-spaces. For this, we design two new loss functions to capture unique relation-specific properties for antonyms and synonyms, i.e., symmetry, transitivity and the special *trans-transitivity* (explained in section 3.3), in two different sub-spaces namely: SYN for synonyms and ANT for antonyms. Finally, a classifier is trained using features constructed from the distilled information, in addition to other word-level features, to distinguish antonym and synonym pairs.

Note that our problem formulation is same as that of existing works with the distinction that we replace the requirement of a large-scale corpus by the availability of pre-trained word embeddings. This makes our setting more appealing and flexible, as pre-trained embeddings are widely available (e.g., word2vec, Glove, and ELMo), and they have arguably high coverage and quality due to the gigantic train-

ing corpus. In addition, they are available for many languages (Joulin et al. 2016), and are easily adaptable as one can customize the pre-trained embeddings by further training with domain-specific corpus (Rothe and Schütze 2016; Lampinen and McClelland 2017; Vulic and Mrksic 2018). We summarize the major contributions of this paper as follows:

- We propose a novel model for the antonym/synonym distinction. Compared with existing research, our model makes less stringent data requirements (only requiring pre-trained embeddings), hence it is more practical and efficient.
- We introduce Distiller: a set of non-linear encoders to distill task-specific information from pre-trained embeddings in a performance-enhanced fashion. In addition, we propose new loss functions to enforce the distilled representations to capture relation-specific properties.
- We demonstrate the effectiveness of the proposed model by comprehensive experimentation. Our model outperforms the existing research on antonym/synonym distinction by a large margin.
- We construct a new dataset for antonym/synonym distinction task in Urdu language to demonstrate the language-agnostic properties of the proposed model.

2 Related Work

Existing research on antonym/synonym distinction can be classified as (i) embeddings based and (ii) pattern based approaches.

Embeddings Based Approaches The embeddings based approaches rely on the distributional hypothesis, i.e., *the words with similar (or opposite) meanings appear in a similar context* (Turney and Pantel 2010). These models are trained using neural networks (Mikolov et al. 2013a; 2013b) or matrix factorization (Pennington, Socher, and Manning 2014). Dominant embeddings based approaches rely on training embedding vectors using different features extracted from large scale text corpora. For example, Scheible, Schulte im Walde, and Springorum (2013) explained the distributional differences between antonyms and synonyms. Adel and Schütze (2014) employed co-reference chains to train skip-gram model to distinguish antonyms. Nguyen, Schulte im Walde, and Vu (2016) integrated distributional lexical contrast information in the skip-gram model for antonym and synonym distinction.

The supervised variants of the embeddings based models employ existing resources, i.e., thesaurus in combination with the distributional information for the distinction task. Pham, Lazaridou, and Baroni (2015) introduced multi-task lexical contrast by augmenting the skip-gram model with supervised information from WordNet. Ono, Miwa, and Sasaki (2015) proposed a model that uses distributional information alongside thesaurus to detect probable antonyms. The major limitation of the embeddings based methods is their inability to discriminate between different lexico-semantic relations; e.g., in Glove the top similar words for the word `small` yield a combination of synonyms (e.g., `tiny`, `little`), antonyms (e.g., `large`, `big`), and irrelevant terms (e.g., `very`, `some`).

Pattern Based Approaches The pattern based approaches rely on capturing lexico-syntactic patterns from large scale text corpora. For example, Lin et al. (2003) proposed two patterns, i.e., *either X or Y* and *from X to Y* particularly indicative of antonym pairs. Roth and Schulte im Walde (2014) used discourse markers in combination with the lexico-syntactic patterns to distinguish antonyms from synonyms. Schwartz, Reichart, and Rappoport (2015) used symmetric patterns to assign different vector representations to the synonym and antonym pairs. Nguyen, Schulte im Walde, and Vu (2017) integrated the lexico-syntactic pattern features with the distributional information.

A major limitation of the pattern based methods is the huge overlap in feature space (i.e., lexico-syntactic patterns) across different classes, which hinders improvement in performance. For example, a commonly confused pattern is the noun-compound, which can be used to represent both the synonym pairs (`government administration`; `card board`) and the antonym pairs (`graduate student`; `client server`).

3 The Proposed Model

3.1 Problem Definition

In this paper, we aim to build a model that can classify a given pair of words as either synonyms or antonyms. We consider synonyms as semantically similar words (having similar meanings), and antonyms as highly contrasting words (having opposite meanings). Similar to the previous works (Roth and Schulte im Walde 2014; Schwartz, Reichart, and Rappoport 2015; Nguyen, Schulte im Walde, and Vu 2017), we assume the availability of training dataset, i.e., pairs of words with class labels indicating the pair satisfies either synonym or antonym relation. In contrast to the existing research, the proposed model no longer relies on a text corpus. We replace it with the available pre-trained word embeddings.

3.2 Overview

Our proposed model consists of two phases: in *Phase I*, we train the Distiller to distill task-specific representations of all words in the pre-trained embeddings via non-linear projections; in *Phase II*, we train a classifier that exploits features constructed from the distilled representations together with other word-level features to classify a given pair of words into either synonyms or antonyms. Both phases are trained in a supervised fashion using the same set of training instances.

We argue that the proposed two-phase design has the following advantages: (i) it allows us to answer the question, *whether we can collect enough information from pre-trained embeddings for synonym/antonym classification*, and (ii) it provides us with the maximal flexibility to accommodate many other types of features, including corpus-level features, in our future work.

3.3 Phase I (Distiller)

Distiller uses two different neural-network encoders to project pre-trained embeddings to two new sub-spaces in a non-linear fashion. Each encoder is a feed-forward neural network with two hidden layers using sigmoid as the activation function. Mathematically, we model them as encoder functions; we use $enc_S(v)$ and $enc_A(v)$ for projections of word v's pre-trained embedding vector to the *synonym and antonym sub-spaces*, respectively.

We observe that antonyms and synonyms are special kind of relations (denoted as r_A and r_S, respectively), that exhibit some unique properties. Synonyms possess *symmetry* and *transitivity*, whereas, antonyms possess *symmetry* and *trans-transitivity*.

Symmetry implies that $r(a, b)$ *if and only if* $r(b, a)$, where $r(a, b)$ means a and b participate in the relation r. For synonyms, the *transitivity* implies that *if* $r_S(a, b)$ *and* $r_S(b, c)$ *then* $r_S(a, c)$ *also holds*. For antonyms, the *trans-transitivity* implies that *if* $r_A(a, b)$ *and* $r_S(b, c)$ *then* $r_A(a, c)$ — as shown by the inferred antonym relation between *good* and *evil* in Figure 1(a). Trans-transitivity is a unique property that helps to infer probable antonym pairs.

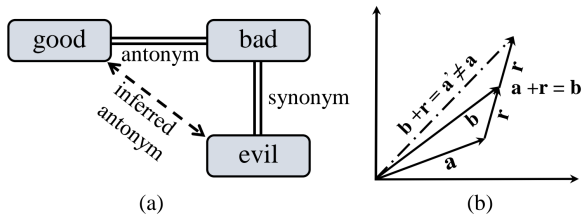

Figure 1: (a) Illustrating the trans-transitivity of antonym and synonym relations, (b) Limitation of the translational embeddings to capture symmetric relations

We note that the existing translational entity embedding models (Bordes et al. 2013; Yoon et al. 2016) are not an appropriate choice for this problem. In translational embedding models, a relation r is modeled as a translation vector $\mathbf{r}$, and $r(a, b)$ implies the L_1 or L_2-norm of $(\mathbf{a} + \mathbf{r} - \mathbf{b})$ is small. As shown in Figure 1(b), it is not possible for the translational embeddings to preserve symmetry by accommodating both vector operations $\mathbf{a} + \mathbf{r} = \mathbf{b}$ and $\mathbf{b} + \mathbf{r} = \mathbf{a}$ at the same time.

Formally, if we try to model a symmetric relation (i.e., synonyms or antonyms) via the translational model, we have

$$\left.\begin{array}{l} \mathbf{a} + \mathbf{r} - \mathbf{b} = \boldsymbol{\epsilon}_1 \\ \mathbf{b} + \mathbf{r} - \mathbf{a} = \boldsymbol{\epsilon}_2 \end{array}\right\} \Rightarrow \quad 2\mathbf{r} = \boldsymbol{\epsilon}_1 + \boldsymbol{\epsilon}_2$$

It means $\mathbf{r}$ must be small to be able to model both $r(a, b)$ and $r(b, a)$. This leads to huge challenge in distinguishing different $\mathbf{r}$s, or has the risk of modeling one of the two pairs badly. A similar explanation suffices to expose the limitation of translational embeddings to preserve transitivity.

While the translational embeddings fail to capture the antonym and synonym relation pairs, we propose a model that:

- for each word vector, creates two distilled embeddings in two sub-spaces (i.e., SYN and ANT).
- models the symmetry and transitivity preserving synonym relation in SYN.
- models the symmetry and trans-transitivity preserving antonym relation using both ANT and SYN.

Loss Function for Synonyms In order to capture synonym relations in SYN, we use a margin based loss to embed the vectors corresponding to synonym pairs (positive examples) close to each other, while at the same time pushing apart the vectors for irrelevant pairs (negative examples). This formulation preserves the symmetry and the transitivity of synonym pairs in the SYN sub-space. While the symmetry is enforced by the commutative nature of the inner product of real vectors[1], the transitivity is preserved by following justification: if $\mathbf{a}$ is embedded close to $\mathbf{b}$ and $\mathbf{b}$ is embedded close to $\mathbf{c}$, then $\mathbf{a}$ has a high chance of being embedded close to $\mathbf{c}$. It, moreover, ensures that we can distinguish between the synonym pairs and the irrelevant pairs within the SYN sub-space.

The loss function for modeling the synonyms is shown in Equation (1). Note that only the embeddings corresponding to the SYN sub-space are involved.

$$\begin{aligned} L_S = & \sum_{(a,b)\in T_S} \mathsf{ReLU}(1 - f(a, b)) \\ & + \sum_{(a',b')\in T'_S} \mathsf{ReLU}(1 + f(a', b')) \end{aligned} \tag{1}$$

where $f(a, b) = \tanh(\langle enc_S(a), enc_S(b)\rangle)$, T_S is the set of synonym training instances, T'_S is the set of negative pairs, and $enc_S(v)$ is the non-linear mapping for the words in the vocabulary V to distilled embedding in the SYN sub-space.

T'_S is generated from T_S by repeating the following corruption process $k = 5$ times: randomly pick a or b from T_S, and replace it with another randomly sampled word from training vocabulary.

Loss Function for Antonyms In order to capture the symmetry and trans-transitivity for antonym pairs, our idea is to *correlate* the ANT and SYN sub-spaces.

For illustration, consider an antonym pair $r_A(a, b)$, and let b_i be any synonym of b. When the fact that b_i and b are synonyms is sufficiently learned in SYN, we expect $\mathbf{b}_i$ to be close to $\mathbf{b}$, later, we encourage the embedding vector $\mathbf{a} \in$ ANT to be close to $\mathbf{b} \in$ SYN. As a result, it is highly likely that $\mathbf{a} \in$ ANT will also be close to $\mathbf{b}_i \in$ SYN.

In order to capture the trans-transitivity using the Distiller, for a given antonym relation pair $r_A(a, b)$, we bring the vector $\mathbf{a} \in$ ANT close to the vector $\mathbf{b} \in$ SYN. For symmetry, we augment the training data by swapping the relation pairs $r_A(a, b)$ to get $r_A(b, a)$, and correspondingly update the embeddings for the vectors $\mathbf{b} \in$ ANT and $\mathbf{a} \in$ SYN. This design preserves the symmetry and the trans-transitivity property for the antonym pairs using both SYN and ANT subspaces. The loss function for modeling antonyms is illustrated in Equation (2). Note that unlike Equation (1), for this

[1] Inner product can be regarded as some notion of similarity.

loss function, the embeddings for both the sub-spaces are involved.

$$L_A = \sum_{(a,b)\in T_A} \mathsf{ReLU}(1 - g(a,b)) + \sum_{(a',b')\in T'_A} \mathsf{ReLU}(1 + g(a',b')) \quad (2)$$

where $g(a,b) = \tanh(\langle enc_A(a), enc_S(b)\rangle)$, T_A is the set of the antonym training instances, T'_A is the set of negative pairs, $enc_S(v)$ is defined as in Equation (1) and $enc_A(v)$ is the non-linear mapping for the words in the vocabulary V to distilled embedding in the ANT sub-space.

T'_A is generated from T_A by repeating the following corruption process $k = 5$ times: randomly pick a or b from T_A, and replace it with another randomly sampled word from the training vocabulary.

Synonymy and Antonymy Scores For a given word pair (a, b), we use our distilled embeddings to compute its *synonymy score* as $cos(enc_S(a), enc_S(b))$ and the *antonymy score* as $\max(\cos(enc_A(a), enc_S(b)), \cos(enc_A(b), enc_S(a)))$.

Loss Function for Distiller The loss function of the Distiller is $L_S + L_A + L_M$, where L_S and L_A have been defined in Equations (1) and (2) respectively. L_M is the cross-entropy loss defined in Equation (3).

$$L_M = -\frac{1}{N}\sum_{i=1}^{N} \log(\hat{p}(y_i|a_i, b_i)) \quad (3)$$

For L_M, we concatenate the synonymy score $\mathbf{x}_1$ and the antonymy score $\mathbf{x}_2$ from the distilled embeddings to form the feature vector $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]$, and use softmax function (Equation (4)) to predict the class label.

$$\hat{p}(\mathbf{y} \mid (\mathbf{a}, \mathbf{b})) = softmax(\mathbf{W}\mathbf{x} + \hat{\mathbf{b}})$$
$$\hat{y} = \underset{y}{\operatorname{argmax}}\, \hat{p}(\mathbf{y}|\mathbf{a}, \mathbf{b}) \quad (4)$$

where $\mathbf{W}$ is the weight matrix and $\hat{\mathbf{b}}$ is the bias. We optimize the loss of the Distiller, in an end-to-end fashion.

3.4 Phase II (Classifier)

In Phase II, we train a classifier to distinguish the synonym and antonym pairs. We use the XGBoost classifier (Chen and Guestrin 2016), and employ following features.

New Features from Distiller For a given test pair (a, b), we compute two new features exploiting our distilled embeddings (1) *synonymy score* and (2) *antonymy score*.

Features from Pre-trained Embeddings Similar to the best performing solutions for antonym and synonym distinction that consider a combination of lexico-syntactic patterns and the distributional embeddings, we use distributional similarity score, i.e., $cos(\mathbf{a}, \mathbf{b})$ as a classification feature, where $\mathbf{a}$ and $\mathbf{b}$ correspond to the *pre-trained* word embedding vectors. It helps in capturing highly confident antonym and/or synonym pairs trained using distributional information. In Table 3, this feature is labeled as: *distributional*.

Features from Negation Prefix Words that differ only by one of the known negation prefixes are highly unlikely to be synonym pairs, e.g., `(able,unable)`, `(relevant,irrelevant)` etc. Similar to Rajana et al. (2017), we use the following set of negation prefixes: {*de, a, un, non, in, ir, anti, il, dis, counter, im, an, sub, ab*}. We use a binary feature for the candidate pairs, where the feature value is 1 only if the two words in the candidate pair differ by one of the negation prefix. In Table 3, this feature is labeled as: *prefix*.

4 Experiments

4.1 Dataset

For model training, we use an existing dataset previously used by (Schwartz, Reichart, and Rappoport 2015; Roth and Schulte im Walde 2014; Nguyen, Schulte im Walde, and Vu 2016; 2017). It has been accumulated from different sources encompassing WordNet (Fellbaum 1998) and WordNik[2]. The details of the dataset are given in Table 1, it contains antonym and synonym pairs for three categories (i.e., verbs, adjectives and nouns) in the ratio (1:1). In order to come up with a unanimous platform for comparative evaluation, we use the priorly defined data splits by the existing models to training, test and dev sets. The training data is used to train the Distiller in Phase-I and the classifier in Phase-II. The development data is used for Distiller's parameter tuning. The model performance is reported on the test set.

Category	Train	Dev	Test	Total
Verb	2534	182	908	3624
Noun	2836	206	1020	4062
Adjective	5562	398	1986	7946

Table 1: Antonym/Synonym distinction dataset

4.2 Experimental Settings

One advantage of our model is its ability to work with any set of word embeddings. For experimentation, we use following set of pre-trained embeddings:

1. **Random Vectors** We use 300d random vectors.
2. **Glove** (Pennington, Socher, and Manning 2014), purely unsupervised word embeddings. We use 300d pre-trained Glove embeddings.
3. **dLCE** (Nguyen, Schulte im Walde, and Vu 2016), customized embeddings for antonym synonym distinction task. Its dimensionality is 100d.
4. **ELMO** (Peters et al. 2018), deep contextualized embeddings based on character sequences. Its dimensionality is 1024d.

[2]https://www.wordnik.com/

Note that our model does not rely on a text corpus, so for the ELMO, we only consider the character sequence of the words in our dataset to acquire the embedding vectors. For Glove and dLCE, the vectors corresponding to the out-of-vocabulary tokens were randomly initialized. We use a smaller and compact representation for the sub-spaces, as they are supposed to preserve only the task-specific information. The dimensionality of each sub-space, (i.e., ANT and SYN) is set to 60d. The neural network encoders used in the Distiller employ 80 units in the first layer and 60 units in the second layer. We use the Adam-Optimizer (Kingma and Ba 2014) to train the Distiller. All the experiments are performed on Intel Xenon Xeon(R) CPU E5-2640 (v4) with 256 GB main memory and Nvidia 1080Ti GPU.

4.3 Baseline Models / Model Comparison

We compare our work against the following models.

Direct Baseline. The Direct baseline is used to show the performance of using only the pre-trained embeddings to distinguish the antonym and synonym pairs. For Direct baseline, we cluster the vector difference for antonym and synonym pairs using k-means clustering to get k-pivot vectors as representatives of synonym and/or antonym candidate pairs. For classification, we use an XGBoost classifier with (i) cosine similarity, (ii) distance to the pivot vectors and (iii) vector difference as features.

Discourse Markers. Roth and Schulte im Walde (2014) used discourse markers (indicators of relations) alongside lexico-syntactic patterns to design vector space models. For the given data set, Michel Roth has already computed the performance of his methods (Nguyen, Schulte im Walde, and Vu 2017), we use the same scores for comparison.

Symmetric Patterns. Schwartz, Reichart, and Rappoport (2015) used automated techniques to extract symmetric patterns (sequence of 3-5 tokens encompassing two wild-cards and 1-3 tokens) from plain text. We use 500d embeddings (from author's web-page [3]). Similar to (Nguyen, Schulte im Walde, and Vu 2017), we calculate cosine similarity between the test pair and use an SVM classifier to categorize them.

AntSynNET. Nguyen, Schulte im Walde, and Vu (2017) proposed two different architectures: (i) AntSynNET: a pattern-based model and (ii) Combined AntSynNET, i.e., combining the pattern-based model with the distributional embeddings. The Combined AntSynNET comprises two models namely: (1) *AntSynNET + Glove* and (2) *AntSynNET + dLCE*, acquired by using Glove/dLCE embeddings. We use the scores reported in the published paper.

4.4 Main Results

The results for the proposed model are shown in Table 2. Here, we use two different sets of pre-trained embeddings for Distiller, i.e., Glove and dLCE. The results are accordingly compared with the corresponding baseline models using the same settings.

[3]https://homes.cs.washington.edu/~roysch/papers/sp_embeddings/sp_embeddings.html

Comparing the results for the Glove embeddings, our model outperforms the Direct baseline and previously best performing models for all three classes by a significant margin. We observe that for nouns the improvement in performance is relatively lower as compared with the verbs and adjectives. It is due to the effect of polysemy, which is more dominant among nouns. The unsupervised embeddings are unable to handle polysemous words and mostly the embedding vectors are oriented in the direction of most common sense (as explained in detail in the section 4.8). This finding is also aligned with that of (Scheible, Schulte im Walde, and Springorum 2013), which states that synonym and antonym pairs conforming to verbs and adjectives have relatively high contextual clues compared with that of nouns. Overall results for our model with Glove embeddings show that even getting started with entirely unsupervised embeddings having a mixture of different lexical-semantic relations, the Distiller is able to distill the task-specific information and outperform the previous state-of-the-art models by a large margin.

Especially noteworthy is the performance of our model with Distiller trained on dLCE. It simply outperforms the best performing model trained using dLCE, yielding a much higher value of F1 across all three word classes. Compared with the previous state-of-the-art, it improves the F1 score by 18%, 17% and 6% for adjectives, verbs and nouns respectively. This drastic improvement in performance explains that in contrary to the Glove embeddings that contains a blend of lexical-semantic information, the dLCE embeddings are enriched with more distinguishing information and the Distiller helps to effectively distill the most relevant task-specific information for antonym/synonym distinction.

Such promising results strengthen our claim that the pre-trained embeddings contain a blend of lexico-semantic information and we can acquire task-specific information by projecting them to low-dimensional subspaces.

4.5 Detailed Performance of Our Model

Table 3 shows the result of our model with different set of features and pre-trained embeddings. We also report the performance of our model with random vectors in order to analyze the performance gained by the unsupervised contextual embeddings.

While the Distiller contributes the most distinction power to our model, we also analyze the results of applying either the *distributional* feature or the *prefix* feature, as well as both of them. It can be seen that the *prefix* feature slightly improved the performance for nouns, yielding a higher F1 score, whereas, for adjectives and verbs, the increase in performance was not so significant. It confirms that the distilled embeddings have a strong distinction power and even the best known features for antonymy detection, i.e., *prefix*, had a very little impact on overall performance. The *distributional* feature slightly improved the performance of the nouns, however, it deteriorated the performance of the verbs and the adjectives. For most cases, using all the features at the same time (i.e., Distiller + *distributional* + *prefix*) had a complementary effect on the model performance, it helped the model to reinforce the decision for confident candidate

Embeddings	Model	Adjective			Verb			Noun		
		P	R	F1	P	R	F1	P	R	F1
None	Discourse Markers	0.717	0.717	0.717	0.789	0.787	0.788	0.833	0.831	0.832
	Symmetric Patterns	0.730	0.706	0.718	0.560	0.609	0.584	0.625	0.393	0.482
	AntSynNET	0.764	0.788	0.776	0.741	0.833	0.784	0.804	0.851	0.827
Glove	Direct Baseline	0.700	0.619	0.657	0.634	0.630	0.632	0.682	0.647	0.664
	AntSynNET	0.750	0.798	0.773	0.717	0.826	0.768	0.807	0.827	0.817
	Our-Model	**0.854**	**0.917**	**0.884**	**0.871**	**0.912**	**0.891**	**0.823**	**0.866**	**0.844**
dLCE	Direct Baseline	0.897	0.897	0.897	0.857	0.833	0.845	0.890	0.859	0.874
	AntSynNET	0.763	0.807	0.784	0.743	0.815	0.777	0.816	0.898	0.855
	Our-Model	**0.912**	**0.944**	**0.928**	**0.899**	**0.944**	**0.921**	**0.905**	**0.918**	**0.911**

Table 2: Antonym/Synonym distinction performance comparison against baseline models

pairs.

Note that the Distiller trained using random vectors results in a significant reduction in performance (first row in Table 3). It shows that the Distiller is not governed by input bias, rather it distills information contained in unsupervised contextual embeddings. Overall results in Table 3 show that our model outperforms the Direct baseline and the previous state-of-the-art model (AntSynNET) for all sets of pre-trained embeddings. These results, moreover, confirm that the dLCE embeddings are indeed customized for antonym/synonym distinction task and on contrary to the AntSynNET, the proposed framework along with the Distiller is able to use this information more effectively.

Recognizing Irrelevant Pairs Although, our current problem setting is not explicitly designed for multi-class classification, our model has an implicit ability to recognize irrelevant pairs. For irrelevant pairs, the resultant projections of words in SYN and ANT sub-spaces end up in a narrow region far from their synonyms and antonyms. For analysis, we augmented the dataset by adding an equal proportion of randomly selected words as "irrelevant" pairs and retrained the classifier in phase II for multi-class classification. With Glove embeddings, our model can distinguish three classes (antonyms, synonyms and irrelevant pairs) with F1 = 0.813, 0.775, and 0.818 for adjectives, verbs, and nouns respectively. These scores are better than the binary classification in previous state-of-the-art (i.e. F1= 0.773, 0.768 and 0.817 for AntSynNET), shown in Table 2.

4.6 Training Time

We analyzed the pre-processing and running time of our model in comparison with the previous best performing pattern-based approach, i.e., AntSynNET.

Our model doesn't require any pre-processing, whereas, on our machine (explained in section 4.2) the AntSynNET takes more than 1 week to parse each sentence in the wiki dump. Moreover, our model is more than 100 times faster to train (e.g., AntSynNET takes more than 50 hours for model training, whereas, our model takes less than half an hour).

4.7 Language-Agnostic Model

We verify the language agnostic property of our model by testing its performance for antonym/synonym distinction task on another language. We manually curated a dataset for the Urdu language using linguistic resource and expert verification. It consists of 750 instances with an equal proportion of antonyms and synonyms. We split the dataset into 70% train, 25% test and 5% validation set. We use fasttext for Urdu language (Joulin et al. 2016) as the pre-trained embeddings. The performance of our model for the Urdu language is shown in Table 4. Huge improvement in the performance compared with the Direct baseline shows that the proposed model is language-agnostic and has potential to work for other languages provided with the availability of pre-trained embeddings and a minimal set of training seeds.

4.8 Analyses of Errors and Distilled Embeddings

In this section, we perform a detailed error analysis, followed by analyzing the probable antonyms/synonyms captured via distilled embeddings. We also list the key advantages of the Distiller.

Error Cases We randomly selected 50 error cases from our model with Glove embeddings for analysis. We categorized these errors into three distinct categories: (i) 76% errors are caused by the polysemous words (ii) 8% errors are caused by the rarely used words (iii) 16% other errors.

A major portion of these errors are caused by the inherent limitation of the unsupervised embeddings, i.e., its inability to deal with multiple senses and rare tokens. For case (i), we analysed synonym pairs, e.g., `(author, generator)`, and antonym pairs, e.g., `(frog, english)`. Here, at least one word is polysemous with embeddings oriented in direction of common sense, which is different from the sense required in antonym/synonym relation. This phenomenon is more dominant for the nouns. For example, `generator`'s vector is close to electrical device; `frog` is commonly used for reptile, however, in the given pair it represents a French person. For case (ii), the embeddings corresponding to the rarely used tokens are not adequately trained, e.g., `natal` is a rarely used word for `native`. This results the relation pair to have very low embeddings' similarity (orthogonal vectors), which limits the Distiller to capture meaningful projections to low-dimensional sub-spaces.

We also observe that increase in word frequency has a deteriorating effect on model performance. This is because most of the high frequency words are polysemous. For analysis, we evenly partitioned the test data (word pairs (a, b)) based on the minimum frequency of two words in a large-scale text corpus. For the least frequent 10% adjective word pairs, the F1 is 0.914, and for the most frequent 10% word

Embeddings	Model	Adjective			Verb			Noun		
		P	R	F1	P	R	F1	P	R	F1
Random Vectors	Distiller + *distributional* + *prefix*	0.639	0.769	0.698	0.719	0.833	0.771	0.672	0.736	0.702
Glove	Direct Baseline	0.700	0.619	0.657	0.634	0.630	0.632	0.682	0.647	0.664
	AntSynNET	0.750	0.798	0.773	0.717	0.826	0.768	0.807	0.827	0.817
	Distiller	**0.859**	0.912	**0.885**	0.866	**0.914**	0.889	**0.823**	0.848	0.835
	Distiller + *distributional*	0.852	0.914	0.884	**0.873**	0.910	**0.891**	0.821	0.853	0.837
	Distiller + *prefix*	0.855	0.916	0.884	0.862	0.913	0.887	0.822	**0.868**	**0.844**
	Distiller + *distributional* + *prefix*	0.854	**0.917**	0.884	0.871	0.912	**0.891**	**0.823**	0.866	**0.844**
dLCE	Direct Baseline	0.897	0.897	0.897	0.857	0.833	0.845	0.890	0.859	0.874
	AntSynNET	0.763	0.807	0.784	0.743	0.815	0.777	0.816	0.898	0.855
	Distiller	**0.920**	0.938	0.929	**0.904**	0.930	0.917	0.902	0.909	0.906
	Distiller + *distributional*	0.905	0.945	0.924	0.886	0.938	0.911	0.893	0.918	0.905
	Distiller + *prefix*	0.914	**0.947**	**0.930**	0.893	**0.944**	0.918	0.894	**0.925**	0.909
	Distiller + *distributional* + *prefix*	0.912	0.944	0.928	0.899	**0.944**	**0.921**	**0.905**	0.918	**0.911**
ELMO	Direct Baseline	0.676	0.589	0.63	0.661	0.665	0.663	0.673	0.633	0.653
	Distiller	**0.839**	0.896	0.866	**0.871**	0.926	0.898	**0.833**	0.869	0.850
	Distiller + *distributional*	0.835	0.898	0.866	0.870	0.929	0.899	0.831	0.875	0.853
	Distiller + *prefix*	0.835	**0.909**	**0.871**	0.868	0.930	0.898	0.831	0.876	0.852
	Distiller + *distributional* + *prefix*	**0.839**	0.905	**0.871**	0.869	**0.933**	**0.900**	0.832	**0.884**	**0.857**

Table 3: Performance of proposed model under different settings

Model	P	R	F1
Direct baseline	0.623	0.533	0.575
Our Model	**0.897**	**0.867**	**0.881**

Table 4: Antonym/Synonym distinction for Urdu language

pairs, the F1 drops down to 0.820. A similar trend is observed for nouns and verbs.

Effectiveness of Distiller Pre-trained word embeddings yield a blend of lexical-semantic relations as nearest neighbor of an input word. On contrary, the Distiller provides us the provision to separately analyze just the antonyms and/or synonyms of a given word. We analyzed the nearest neighbors of `large` using Glove and the distilled embeddings. Results corresponding to Glove embeddings (first column in Table 5) show that it makes no distinction among lexico-semantic relations, e.g., the antonym `small` is ranked highest in the list. The results corresponding to the distilled embeddings (column-2) show that the top ranked terms (e.g., `immense, gigantic`) are indeed the synonyms. Likewise, the results in column-3 show that the top ranked terms by the Distiller (e.g., `slender, thin`) are indeed the antonyms.

Glove	Distiller(Synonyms)	Distiller(Antonyms)
small (antonym)	immense	slender
larger (synonym)	gigantic	thin
smaller (antonym)	influential	lightweight
huge (synonym)	full	small
sized (other)	enormous	inconspicuous

Table 5: Top-5 nearest neighbors of the word: `large` using Glove embeddings vs synonyms and antonyms captured by the distilled embeddings

Key Advantages of Distiller The key advantages of Distiller compared with the previous state-of-the-art pattern based methods are listed below:

- Distiller is not constrained by naive assumptions. For example, the pattern based methods work only if a dependency path exists between candidate antonym/synonym pair, and it is impossible for every feasible synonym and/or antonym pair to co-occur within a sentence. Likewise, the Distiller is not limited by the challenges posed by the overlapping nature of lexico-syntactic patterns, which hinder the performance of pattern based methods.
- Distiller allows explicitly constraining the sub-spaces. This formulation is best suited for lexico-semantic analysis, as it allows appropriate re-ordering of the sub-spaces to capture relation-specific characteristics.
- Thanks to the wide availability of pre-trained embeddings with various improvements and for most languages, Distiller is a flexible and efficient choice for lexical-semantic analysis requiring no pre-processing and little linguistic knowledge.

5 Conclusions and Future Work

In this paper, we proposed a novel framework, Distiller, for antonym and synonym distinction. It employs carefully crafted loss functions to project the pre-trained embeddings to low-dimensional task-specific sub-spaces in a performance enhanced fashion. Results show that the Distiller outperforms previous methods on the antonym and synonym distinction task. In the future, we will extend the proposed framework to other lexical-semantic relations (e.g., hypernymy detection) and to other embedding models such as the hyperbolic embeddings (Nickel and Kiela 2017).

Acknowledgments. This research was partially funded by ARC DPs 170103710 and 180103411, and D2DCRC DC25002 and DC25003, and NSFC Grant 61872446.

References

Adel, H., and Schütze, H. 2014. Using mined coreference chains as a resource for a semantic task. In *EMNLP*, 1447–1452. ACL.

Bordes, A.; Usunier, N.; García-Durán, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, 2787–2795.

Chen, T., and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *KDD*, 785–794. ACM.

Fellbaum, C. 1998. *WordNet*. Wiley Online Library.

Glavas, G., and Vulic, I. 2018. Discriminating between lexico-semantic relations with the specialization tensor model. In *NAACL-HLT (2)*, 181–187. Association for Computational Linguistics.

Joulin, A.; Grave, E.; Bojanowski, P.; Douze, M.; Jégou, H.; and Mikolov, T. 2016. Fasttext.zip: Compressing text classification models. *CoRR* abs/1612.03651.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Lampinen, A. K., and McClelland, J. L. 2017. One-shot and few-shot learning of word embeddings. *CoRR* abs/1710.10280.

Lin, D.; Zhao, S.; Qin, L.; and Zhou, M. 2003. Identifying synonyms among distributionally similar words. In *IJCAI*, 1492–1493. Morgan Kaufmann.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

Mohammad, S.; Dorr, B. J.; Hirst, G.; and Turney, P. D. 2013. Computing lexical contrast. *Computational Linguistics* 39(3):555–590.

Nguyen, K. A.; Schulte im Walde, S.; and Vu, N. T. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *ACL (2)*. The Association for Computer Linguistics.

Nguyen, K. A.; Schulte im Walde, S.; and Vu, N. T. 2017. Distinguishing antonyms and synonyms in a pattern-based neural network. In *EACL (1)*, 76–85. Association for Computational Linguistics.

Nickel, M., and Kiela, D. 2017. Poincaré embeddings for learning hierarchical representations. In *NIPS*, 6341–6350.

Ono, M.; Miwa, M.; and Sasaki, Y. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *HLT-NAACL*, 984–989. The Association for Computational Linguistics.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*, 1532–1543. ACL.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *NAACL-HLT*, 2227–2237. Association for Computational Linguistics.

Pham, N. T.; Lazaridou, A.; and Baroni, M. 2015. A multitask objective to inject lexical contrast into distributional semantics. In *ACL (2)*, 21–26. The Association for Computer Linguistics.

Rajana, S.; Callison-Burch, C.; Apidianaki, M.; and Shwartz, V. 2017. Learning antonyms with paraphrases and a morphology-aware neural network. In **SEM*, 12–21. Association for Computational Linguistics.

Roth, M., and Schulte im Walde, S. 2014. Combining word patterns and discourse markers for paradigmatic relation classification. In *ACL (2)*, 524–530. The Association for Computer Linguistics.

Rothe, S., and Schütze, H. 2016. Word embedding calculus in meaningful ultradense subspaces. In *ACL (2)*. The Association for Computer Linguistics.

Scheible, S.; Schulte im Walde, S.; and Springorum, S. 2013. Uncovering distributional differences between synonyms and antonyms in a word space model. In *IJCNLP*, 489–497. Asian Federation of Natural Language Processing / ACL.

Schwartz, R.; Reichart, R.; and Rappoport, A. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *CoNLL*, 258–267. ACL.

Turney, P. D., and Pantel, P. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Intell. Res.* 37:141–188.

Vulic, I., and Mrksic, N. 2018. Specialising word vectors for lexical entailment. In *NAACL-HLT*, 1134–1145. Association for Computational Linguistics.

Vulic, I. 2018. Injecting lexical contrast into word vectors by guiding vector space specialisation. In *Rep4NLP@ACL*, 137–143. Association for Computational Linguistics.

Yoon, H.; Song, H.; Park, S.; and Park, S. 2016. A translation-based knowledge graph embedding preserving logical property of relations. In *HLT-NAACL*, 907–916. The Association for Computational Linguistics.