Detecting Semantic Difference using Word Embeddings

Alexander Zhang and **Marine Carpuat**
Department of Computer Science
University of Maryland
College Park, MD 20742, USA
alexz@umd.edu, marine@cs.umd.edu

Abstract

Semantic difference is a harder problem than the well-studied semantic similarity, because it tries to detect directional relationships. The task is the following: given word triple (w_1, w_2, d), is d a discriminating attribute belonging to w_1 but not w_2? Our study aims to determine whether word embeddings can address this challenging task. We cast this problem as a supervised binary classification using only word embedding features. With a gaussian SVM model trained only on validation data, we achieve an F-score of 60%. We also show that cosine similarity features are more effective, both in unsupervised systems (F-score of 65%) and supervised systems (F-score of 67%). This work was submitted for Task 10 of SemEval 2018.

1 Introduction

SemEval-2018 Task 10 (Krebs et al., 2018) offers an opportunity to evaluate word embeddings on a challenging lexical semantics problem. Much prior work on word embeddings has focused on the well-established task of detecting semantic similarity (Mikolov et al., 2013a; Pennington et al., 2014; Baroni et al., 2014; Upadhyay et al., 2016). However, semantic similarity tasks alone cannot fully characterize the differences in meaning between words. For example, we would expect the word car to have high semantic similarity with $truck$ and with $vehicle$ in distributional vector spaces, while the relation between car and $truck$ differs from the relation between car and $vehicle$. In addition, popular datasets for similarity tasks are small, and similarity annotations are subjective with low inter-annotator agreement (Krebs and Paperno, 2016).

The task focuses instead on determining semantic difference: given a word triple (w_1, w_2, d), the task consists in predicting whether d is a discriminating attribute applicable to w_1, but not to w_2. For instance, ($w_1 =$ apple, $w_2 =$ banana, $d =$ red) is a positive example as *red* is a typical attribute of *apple*, but not of *banana*.

This work asks to what extent word embeddings can address the challenging task of detecting discriminating attributes. On the one hand, word embeddings have proven useful for a wide range of NLP tasks, including semantic similarity (Mikolov et al., 2013a; Pennington et al., 2014; Baroni et al., 2014; Upadhyay et al., 2016) and detection of lexical semantic relations, either explicitly by detecting hypernymy, lexical entailment (Baroni et al., 2012; Roller et al., 2014; Turney and Mohammad, 2013), or implicitly using analogies (Mikolov et al., 2013b). On the other hand, detecting discriminating attributes requires making fine-grained meaning distinctions, and it is unclear to what extent they can be captured with opaque dense representations.

We start our study with unsupervised models. We propose a straightforward approach where predictions are based on a learned threshold for the cosine similarity difference between (w_1, d) and (w_2, d), representing words using Glove embeddings (Pennington et al., 2014). We use this unsupervised approach to evaluate the impact of word embedding dimensions on performance.

We then compare the best unsupervised configuration to supervised models, exploring the impact of different classifiers and training configurations. Using word embeddings as features, supervised models yield high F-scores on development data, on the final test set they perform worse than the unsupervised models. Our supervised submission yields an F-score of 60%. In later experiments, we show that using cosine similarity as features

is more effective than directly using word embeddings, reaching an F-score of 67%.

2 Task Data Overview

Dataset	Pos	Neg	Total	d Vocab
$train$	6,591	11,191	17,782	1,292
$validation$	1,364	1,358	2,722	576
$test$	1,047	1,293	2,340	577

Table 1: Dataset statistics for the training and validation set: number of positive examples (Pos), number of negative examples (Neg), total number of examples (total), size of vocabulary for discriminant words d (d Vocab)

For development purposes, we are provided with two datasets: a training set and a validation set, whose statistics are summarized in Table 1.

Word triples (w_1, w_2, d) were selected using the feature norms set from McRae et al. (2005). Only visual discriminant features were considered for d, such as is_green. Positive triples (w_1, w_2, d) were formed by selecting w_2 among the 100 nearest neighbors of w_1 such that a visual feature d is attributable to w_1 but not w_2. Negative triples were formed by either selecting an attribute attributable to both words, or by randomly selecting a feature not attributable to either word.

The distribution of the training and validation sets differ: the validation and test sets are balanced, while only 37% of examples are positive in the training set. In addition, the validation and test sets were manually filtered to improve quality, while training examples are noisier. The data split was chosen to have minimal overlap between discriminant features.

3 Unsupervised Systems

All our models rely on Glove (Pennington et al., 2014), generic word embeddings models, pretrained on large corpora: the Wikipedia and English Gigaword newswire corpora. In addition to capturing semantic similarity with distances between words, Glove aims for vector differences to capture the meaning specified by the juxtaposition of two words, which is a good fit for our task.

Because the discriminant features are distinct between train, validation and test, our systems should be able to generalize to previously unseen discriminants. This makes approaches based on word embeddings attractive, as information about word identity is not directly encoded in our model.

3.1 Baseline

We first consider the baseline approach introduced by Krebs and Paperno (2016) to detect the positive examples, where cs denotes the cosine similarity function:

$$cs(word_1, disc) > cs(word_2, disc) \quad (1)$$

3.2 2-Step Unsupervised System

We refine this baseline with a 2-step approach. Our intuition is that d is a discriminant between w_1 and w_2 if the following two conditions hold simultaneously:

1. w_1 is more similar to d than w_2 by more than a threshold t_{thresh}:

$$cs(w_1, d) - cs(w_2, d) > t_{thresh} \quad (2)$$

2. d is highly similar to w_1:

$$cs(w_1, d) > t_{diverge} \quad (3)$$

The condition in Equation 2 aims at detecting negative examples that share the discriminant attribute, and the condition defined by Equation 3 targets negative examples that share a random discriminant. Thresholds t_{thresh} and $t_{diverge}$ are hyper-parameters tuned on the $train.txt$.

3.3 Results

We evaluate unsupervised systems using word embeddings of varying dimensions on the validation set, and report averaged F-scores. As can be seen in Table 2, increasing the dimension of word embeddings improves performance for both systems, and the 2-step model consistently outperforms the baseline. The best performance is obtained by the 2-step model with 300-dimensional word embeddings. We therefore select these embeddings for further experiments.

Vector Dim	50	100	200	300
baseline	.5765	.5965	.6171	.6183
2-step model	.4034	.6130	.6266	.6312

Table 2: Averaged F-Score across GloVe Dimensions between our 2-step unsupervised system and the baseline from Krebs and Paperno (2016), for word vectors of size 50, 100, 200 and 300.

4 Supervised Systems

We consider a range of binary classifiers that operate on feature representations derived from word embeddings $\vec{w_1}, \vec{w_2}$ and $\vec{d}$. We further tuned systems based on 10-fold cross-validation using the concatenation of the training and validation data.

Feature Representations We seek to capture the difference in meaning between w_1 and w_2 and its relation to the meaning of the discriminant word d. Given word embeddings for each of these words $\vec{w_1}, \vec{w_2}$ and $\vec{d}$, respectively, we therefore construct input features based on various embedding vector differences. We experimented with the concatenation of $\vec{w_1}$, $\vec{w_2}$, $\vec{d}$, $\vec{w_1} - \vec{d}$ and $\vec{w_2} - \vec{d}$. Based on cross-validation performance on training and validation data, we eventually settled on the concatenation of $\vec{w_1} - \vec{d}$ and $\vec{w_2} - \vec{d}$ for the SemEval task, which yields a compact representation of $2D$ features, if D is the embedding dimension.

Binary Classifier We consider a number of binary classification models found in scikit-learn: logarithmic regression (LR), decision tree (DT), naïve Bayes (NB), K nearest neighbors (KNN), and SVM with linear (SVM-L), and Gaussian (SVM-G) kernels. We compare linear combinations of word embeddings to the more complex combinations enabled by non-linear models.

Best Performing Model For our submission to SemEval, we used the refined SVM-G trained on *validation.txt*. There were three input triplets for which one word was out of the vocabulary of the Glove embedding model: random predictions were used for these. This system achieved an F-Score of .6018. This is a substantial drop from the averaged cross-validation F-scores obtained during development which reached F-scores of 0.9318 using cross-validation on the validation and training sets together, and 0.9674 using cross-validation on the training set only. Using the released test dataset, *truth.txt*, we consider various experiments to understand the poor performance of the model.

5 Analysis

5.1 Embedding Selection

We first evaluate our hypothesis that word embeddings that perform well in the unsupervised setting would also, in general, perform well for classification. We vary embedding dimensions keeping the rest of the experimental set-up constant (train on *validation.txt*, evaluate on *truth.txt*).

Table 3 shows the performance of all supervised model configurations and of the 2-step unsupervised system. Increasing the word embedding dimensions improves the performance of the 2-step unsupervised system, as observed during the development phase (Section 3). However, the supervised classifier behaves differently: for several linear classifiers (e.g., LR, DT, SVM-L) the best performance is achieved with smaller word embeddings. For the non-linear SVM used for submission (SVM-G), varying the embedding dimensions has little impact on overall performance. The SVM-G classifier's performance is now on par with the linear classifiers, while it performed better on development data.

The best performance overall is achieved by the unsupervised model, and taken together, the supervised results suggest that the submitted system overfit the validation set, and was not able to generalize to make good predictions on test examples.

5.2 Impact of training on train vs. validation examples

We consider the effect of training only on the *validation.txt* dataset. Earlier, we mentioned that the best-performing model was only trained on *validation.txt*, due to concerns that also training on *train.txt* would reduce performance due to noise. We evaluate the performance of refined SVM-G on *truth.txt* over two additional input training datasets: *train.txt* (Train) and *train.txt* merged with *validation.txt* (Merge), and compare to that trained only on *validation.txt* (Validation).

As expected, the model's performs better on the *truth.txt* dataset when trained on *validation.txt*, both of which were manually curated after being generated (Table 4). The smaller but cleaner set of training examples in *validation.txt* yields better test performance than the 6.5 times larger set of noisy examples in *train.txt*. Combining the two training set improves over using noisy examples alone, but degrades performance compared to clean-only data.

5.3 Performance on Seen vs. Unseen Words

Prior work on detecting hypernymy using supervised approaches showed that supervised classifiers can obtain good performance by memorizing what words are good "prototypical hypernyms"

Model	dim=50			dim=100			dim=200			dim=300		
	F	P	R	F	P	R	F	P	R	F	P	R
LR	.5742	.5750	.5741	.5769	.5788	.5770	.5739	.5741	.5738	.5525	.5525	.5524
DT	.5494	.5498	.5503	.5356	.5357	.5359	.5304	.5311	.5314	.5283	.5290	.5293
NB	.5618	.5674	.5634	.5873	.5999	.5903	.5885	.5904	.5884	.5908	.5972	.5918
KNN	.5640	.5746	.5677	.5677	.5715	.5720	.5738	.5737	.5740	.5537	.5575	.5579
SVM-L	.5769	.5778	.5768	.5847	.5904	.5856	.5781	.5791	.5779	.5364	.5372	.5376
SVM-G	.5901	.5909	.5919	.6098	.6099	.6097	.5924	.5923	.5924	.5995	.6002	.5993
2-step	.5937	.5938	.5947	.6042	.6041	.6044	.6278	.6278	.6290	.6484	.6481	.6490

Table 3: F-Score, Precision and Recall computed on $truth.txt$ for the full range of supervised classification models across different embedding dimensions trained on $validation.txt$. The first 6 row are supervised systems, the last row shows the performance of the unsupervised 2-step system.

Vector Dim.	50	100	200	300
Train	.5000	.4690	.5168	.4681
Merge	.5632	.5546	.5835	.5622
Validation	.5901	.6098	.5924	.5995

Table 4: F-scores obtained on the test set, for various embedding dimensions and training input dataset using the refined SVM-G classifier.

rather than by learning anything about the relation between two words (Levy et al., 2015). In this task, there is a similar concern that the supervised approaches might memorize what word pairs are more indicative of discriminative attributes. To gain some insights on this question, we consider the performance on input triples featuring words that have been previously fed into the model at training time.

The hypothesis is that whether or not the model was previously fed the word should not matter due to the nature of the formed input vectors to the model, $\{x_1 - d, x_2 - d\}$. Because the words are not directly encoded into the model, unlike the full input vector model considered before, when a previously seen word is paired with an unseen discriminant, the input to the model is different, and thus, whether or not a word has been previously seen should not impact classification. To test this, we isolate the 1117 input triples featuring words that were included in the $validation.txt$ dataset.

The gain in performance is small when restricting the evaluation to the seen vocabulary is small (Table 7), which indicates that the classifier does not behave significantly differently for seen and unseen words. However, the overall F-Scores remain low.

Vector Dim.	50	100	200	300
Seen-only	.6017	.6148	.6002	.6070
Full	.5901	.6098	.5924	.5995

Table 5: F-score on test triples featuring previously-seen words against full test set

Vector Dim.	50	100	200	300
V1-LR	.6083	.6076	.6369	.6526
V1-KNN	.6045	.6115	.6335	.6587
V1-SVMG	.6039	.6227	.6479	.6681
V2-LR	.6398	.6475	.6463	.6490
V2-KNN	.5376	.5239	.5334	.5221
V2-SVMG	.6304	.6435	.6592	.6598
V3-LR	.6203	.6108	.6167	.6193
V3-KNN	.5356	.5182	.5116	.5308
V3-SVMG	.6099	.6233	.6269	.6309
V4-LR	.6089	.6072	.6378	.6525
V4-KNN	.6088	.6120	.6402	.6589
V4-SVMG	.6102	.6239	.6451	.6708

Table 6: F-score for well-performing models of alternative input variant representations

5.4 Feature Variants

Motivated by the good performance of the unsupervised model based on cosine similarity, we consider four feature representations variants for the supervised classifiers,[1]:

$$V1 = [cs(w_1, w_2), cs(w_1, w_d), cs(w_2, d)]$$
$$V2 = [V1, w_1 - w_2, w_d]$$
$$V3 = [V1, w_1 - w_d, w_2]$$

[1]The KNN, SVM-L, and SVM-G used tuned hyperparameters.

$$V4 = [V1, cs(w_1 - w_2, w_d), cs(w_1 - w_d, w_2)]$$

Variant V1 based only on cosine similarity between all pairs yields competitive F-scores from both the SVM-G and LR models (Table 6), and it competitive with the best-performing unsupervised model. We thus use it as a starting point for subsequent variants. Variants V2 and V3 encode the intuition that we expect $w_1 - w_2 \approx w_d$ and $w_1 - w_d \approx w_2$ for positive examples, and therefore, it is possible that these input representations may perform better than the differences-only model. In doing so, we also risk memorizing actual input words as w_d and w_2 are encoded directly as features. These two variants performed worse than the cosine-only models, suggesting that cosine similarity captures semantic difference better than the high-dimensional word vectors themselves. Also interestingly, the KNN model performed significantly worse in these two variants. The best result is achieved using V4, which augments V1 with cosine features that better capture word relations through embedding differences, with an averaged F-score of .6708 using the SVM-G classifier.

5.5 Cross-Validation Set-up

We further explore why cross-validation scores differed greatly from the final test scores. We constructed initial cross-validation sets using sequential 10% cuts of the training set. This is inconsistent with the actual experimental setup, which had distinct sets of d, the discriminating attribute, between the training and test sets. We experiment segmenting the validation dataset so that each of the cross-validation sets had distinct discriminating attributes. This yields only minor gains (Table 7), suggesting that overfitting to the identity of the discriminating attributes was not an issue.

Vector Dim.	50	100	200	300
V4-KNN	.6050	.6172	.6394	.6574
V4-SVML	.6109	.6042	.6301	.6478
V4-SVMG	.6104	.624	.6404	.6716

Table 7: F-score from well-formed cross-validation sets

6 Conclusion

This study showed the limits of directly using word embeddings as features for the challenging task of capturing discriminative attributes between words. Supervised models based on raw embedding features are highly sensitive to the nature and distribution of training examples. Our Gaussian Kernel SVM overfit the training set and performed worse than unsupervised models that threshold cosine similarity scores on the official evaluation data. Based on this finding, we explore the use of cosine similarity scores as features for supervised classifiers, to capture similarity between word pairs, and between words and word relations as represented by embedding differences. These features turn out to be more useful than directly using the word embedding themselves, yielding our best performing system (F-score of 67%).

While these results are encouraging, it remains to be seen how to best design models and features that capture nuanced meaning differences, for instance by leveraging metrics complementary to cosine and resources complementary to distributional embeddings.

7 Acknowledgements

I offer my sincerest gratitude to my advisor, Dr. Marine Carpuat, for her patience, guidance, and support over the past two years. Her technical and editorial advice were essential in my development as a researcher and the creation of this work. I am grateful to Yogarshi and Xing, for their tutelage during my research.

I would also like to acknowledge a few key people in my academic development: to Dr. Khuller and Dr. Cohen, thank you for giving me direction in my first steps in college, and to Dr. Gasarch, I give my appreciation for turning muffins and my math background into a fulfilling semester-long project.

Thanks to my friends, Alex, Tommy, Jamie, Rex, Will, Stepanov, Patrick, and Jerry to name a few, for supporting me through thick and thin, from all-nighters to Iron Age runs to just listening to my (research) struggles. You guys are amazing.

Finally, thanks to my family, for their dedication and encouragement to pursue the science, and even more for supporting me all this way.

Thank you all for helping compose this chapter of my life.

References

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of EACL 2012*, pages 23–32.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247.

Alicia Krebs, Alessandro Lenci, and Denis Paperno. 2018. Semeval-2018 task 10: Capturing discriminative attributes. In *Proceedings of SemEval-2018: International Workshop on Semantic Evaluation*.

Alicia Krebs and Denis Paperno. 2016. Capturing discriminative attributes in a distributional space: Task proposal. In *RepEval@ACL*.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do Supervised Distributional Methods Really Learn Lexical Inference Relations? In *NAACL HLT 2015*, pages 970–976.

Ken McRae, George S. Cree, Mark S. Seidenberg, and Chris Mcnorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior Research Methods*, 37(4):547–559.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *arXiv Preprint*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic Regularities in Continuous Space Word Representations. In *HLT-NAACL*, volume 13, pages 746–751.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP 2014*, pages 1532–1543.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet Selective: Supervised Distributional Hypernymy Detection. *Proceedings of COLING 2014*, pages 1025–1036.

Peter Turney and Saif Mohammad. 2013. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering*, 1(1):1–42.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual Models of Word Embeddings: An Empirical Comparison. In *Proceedings of ACL*, Berlin, Germany.