ABDN at SemEval-2018 Task 10: Recognising Discriminative Attributes using Context Embeddings and WordNet

Rui Mao, Guanyi Chen, Ruizhe Li and Chenghua Lin
Computing Science Department
University of Aberdeen
Aberdeen, Scotland, UK
{r03rm16, g.chen, ruizhe.li, chenghua.lin}@abdn.ac.uk

Abstract

This paper describes the system that we submitted for SemEval-2018 task 10: capturing discriminative attributes. Our system is built upon a simple idea of measuring the attribute word's similarity with each of the two semantically similar words, based on an extended word embedding method and WordNet. Instead of computing the similarities between the attribute and semantically similar words by using standard word embeddings, we propose a novel method that combines word and context embeddings which can better measure similarities. Our model is simple and effective, which achieves an average F1 score of 0.62 on the test set.

1 Introduction

Capturing discriminative attributes is a novel task, which is very different from classical semantic tasks that model similarities in semantics. The task aims to recognise semantic differences between words. Traditional semantic similarity evaluation tasks were designed for evaluating the quality of word representations based on the fact that words with similar semantics will be close to each other in vector space. Recent state-of-the-art distributed semantic models (Ling et al., 2015; Bojanowski et al., 2017) inspired by the success of word2vec (Mikolov et al., 2013) gave good performance in these similarity measure tasks. Nevertheless, how to capture discriminative attributes between semantically similar words is still a challenge for traditional word embedding methods, because these methods are designed to capture similar semantics.

We have two observations for the nature of the task and the provided data: (1) only limited data is available for model training; (2) the inputs of the model are merely isolated words themselves, which lack context information for applying complex models. Therefore, we propose a novel framework that differentiates two semantically similar words with the attribute word by using their word and context embeddings. We experimented with both Continuous Bag of Words (CBOW) and Skip-gram, demonstrating that using the combination of word and context embeddings outperforms using word embeddings alone.

The contribution of this work can be summarised as follows. We examine word and context embeddings in CBOW and Skip-gram, showing that using both word and context embeddings can better measure the co-occurrence of two words in sentences than simply using word embeddings. Hence our similarity measure can recognise the discriminative attributes of two semantically similar words more accurately.

2 System Description

Our system is trained based on word and context embedding features as well as WordNet features (Fellbaum, 1998). Before introducing our framework in detail, we first introduce the two key technical parts of our framework, i.e., context embedding and WordNet.

2.1 Context embeddings

In contrast to simply using traditional word embeddings which model semantic similarities based on contextual similarities, we consider using both word and context embeddings. Word and context embeddings are the vectors of target words and context words in CBOW and Skip-gram. Using them together can model the co-occurrence of attribute words and distinguished words in a sentence, which is useful in predicting whether the attribute word can distinguish two semantically similar words.

Take the Skip-gram model as an example. Skip-gram uses a neural network with a single hidden

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 1017–1021
New Orleans, Louisiana, June 5–6, 2018.

layer of neurons. Given a target word, the objective function is to maximize the probability of predicting each context word (several words before and after the target word in a training sentence) (Rong, 2014):

$$\arg\max p(w_c|w_t) \tag{1}$$

where w_t is a target word in a training sentence, w_c is a context word of w_t, appearing within the same sentence.

During every training epoch on each context word, the weight matrices before and after the hidden layer will be updated. A row vector in the matrix before the hidden layer is a traditional word embedding v_t, as the vector is updated when the corresponding word w_t is in the target word position. A column vector in the matrix after the hidden layer is defined as context embedding v_c, as the column vector is updated when the corresponding word w_c is in the context word position. Each word has two vectors, v_t and v_c, as each word can be a target word or a context word of other target words. Some popular toolkits, e.g., gensim word2vec (Řehůřek and Sojka, 2010), abandon Skip-gram's context embeddings v_c after training, as experimental research (Nalisnick et al., 2016) proves that simply using v_t or v_c (IN-IN or OUT-OUT in the original paper) for two function or type similar words to measure their similarity yields higher scores.

Actually, the conditional probability in the objective function in Eq. 1 can be expanded as:

$$p(w_c|w_t) = \frac{\exp(v_t \cdot v_c)}{\sum_{i\in|V|}\exp(v_t \cdot v_i)} \tag{2}$$

where V is the vocabulary of the training set, the dot product $v_t \cdot v_c$ in numerator computes the similarity between the target word vector and the context word vector. The denominator is to normalise the similarity into a probability. Thus, given a target word, training the whole model involves updating the matrices before and after the hidden layer to maximize the probability of predicting the context word. This is similar to maximizing the similarity between the target word embeddings v_t and context word embeddings v_c. It means that if we can reuse this trained similarity measure, to compute e.g., cosine similarity, then we will get a much better result. In other words, using both the word and context embeddings of two words that frequently appeared within each other's contexts will result in a better similarity measure, which may be incorporating the co-occurrence information of the two words.

CBOW can be considered as the reverse of Skip-gram. Given a context, the target is to maximize the probability of predicting the target word appearing in the context. Later, we will examine both CBOW and Skip-gram's word and context embeddings in our model.

2.2 Word definition in WordNet

We also introduce features based on word sense definitions in WordNet (Fellbaum, 1998), considering the differences between the definitions of two semantically similar words. The two words may be similar in semantics, but different in definitions. An eligible discriminative attribute word may have high possibility to appear within one of the two word definitions, rather than both of them. For example, *ears* can distinguish *corn* and *broccoli*, as in WordNet, *ears* occurs in the definition of *corn* as *"tall annual cereal grass bearing kernels on large ears: widely cultivated in America in many varieties; the principal cereal in Mexico and Central and South America since pre-Columbian times"*, rather than *broccoli*'s definition that *"plant with dense clusters of tight green flower buds"*. We will also capture such characters to distinguish two words.

2.3 Hypothesis and framework

Our first hypothesis is that an attribute word w_A can distinguish two semantically similar words w_1 and w_2, if the attribute word co-occurs much more frequently with one word than the other in the corpora. In vector space, the attribute word can be closer to a distinguished word than the other one. Our second hypothesis is that if w_A can distinguish w_1 and w_2, w_A may appear within one of the definitions of w_1 and w_2 in WordNet.

The framework of our model can be summarized as: (1) we firstly train word embeddings v_t and context embeddings v_c on a Wikipedia dump. (2) Given a triple (w_1, w_2, w_A), we then compute w_A's cosine similarities with w_1 and w_2, and the difference in their similarities, which are used as three input features in the following classifications. For example, given the context embeddings of w_1 and w_2 and the word embedding of w_A, we compute Feature 1: cosine($v_c^{w_1}, v_t^{w_A}$); Feature 2: cosine($v_c^{w_2}, v_t^{w_A}$); and Feature 3: |cosine($v_c^{w_1}, v_t^{w_A}$) − cosine($v_c^{w_2}, v_t^{w_A}$)|, respec-

1	cosine(w_1, w_A)
2	cosine(w_2, w_A)
3	|cosine(w_1, w_A) − cosine(w_2, w_A)|
4	binary variable, indicating if w_A appears in the WordNet definitions of w_1
5	binary variable, indicating if w_A appears in the WordNet definitions of w_2

Table 1: Feature descriptions.

tively. (3) Next, we introduce two binary features to indicate whether w_A appears in any sense definitions of w_1 and w_2 in WordNet (Feature 4 and 5), respectively. (4) We train a random forest classifier with the above five features (see Table 1) to classify if the attribute word w_A can distinguish two semantically similar words w_1 and w_2.

3 Experimental Settings

Data. The data was provided by the organizers of SemEval 2018 Task 10: Capturing Discriminative Attributes[1]. There are 17,501, 2,722 and 2,340 triples (w_1, w_2, w_A) in training, validation and testing sets, respectively. All the words in the triples are nouns. Note that the discriminative attribute words w_A in the given dataset are selected, because they represent the visual attribute of one of two semantically similar words. For example, *red* can differentiate *apple* and *banana*, because visually, apple is red, while banana is yellow. The task does not consider other discriminative features, such as sound and taste. So, using image features may take advantages in this dataset, however, semantic features can also capture invisible discriminative attributes.

Word and context embedding. We first iteratively train 300 dimensional word and context embeddings based on CBOW and Skip-gram with a Wikipedia dump[2] for 3 epochs respectively, setting a context window of 5 words before and after the target word. Words with frequency less than 5 in the Wikipedia are ignored. The down sampling rate is 10^{-4}.

Based on CBOW and Skip-gram, we test all possible combinations of word and context embeddings to compute cosine similarities. The first combination is context embeddings of the two semantically similar words w_1 and w_2, and word embeddings of the attribute word w_A. In Table 2 and 4, this approach is represented as $v_c^{w1} v_c^{w2} v_t^{wA}$. The second $v_t^{w1} v_t^{w2} v_c^{wA}$ uses word embeddings of w_1 and w_2, and context embeddings of w_A. The third $v_c^{w1} v_c^{w2} v_c^{wA}$ is simply context embeddings of w_1, w_2 and w_A. The fourth $v_t^{w1} v_t^{w2} v_t^{wA}$ is simply word embeddings of w_1, w_2 and w_A.

[1] https://competitions.codalab.org/competitions/17326

[2] https://dumps.wikimedia.org/enwiki/20170920/

4 Results

We cast the challenge task as a supervised classification problem. We first examine which combination of word and context embeddings and which training method (CBOW or Skip-gram) is optimal in this task. In this step, we only use Feature 1-3 (see Table 1) to classify the triple (w_1, w_2, w_A).

As can be seen in Table 2, both the CBOW based methods that use word and context embeddings yield the highest average F1 of 0.55 in the validation set. Skip-gram based models generally perform worse than CBOW based models, but using Skip-gram word embeddings of w_1 and w_2 and context embeddings of w_A also outperforms the word embedding based model in the validation set. The experiments running on the test set show similar trends that word and context embedding based models outperform word embedding based models. Such results demonstrate that using word and context embeddings together can better distinguish two semantically similar words with an attribute word, than simply using standard word embeddings. The results also support our first hypothesis that if the attribute word frequently appears in one word's context than the other one, it can distinguish the two words.

We also examine WordNet definition features individually. As shown in Table 3, simply using Feature 4-5 cannot classify the triple accurately. The F1 score of setting positive label as 1 is very low on the validation set (F1=36%). This is for the reason that an eligible discriminative attribute word cannot always associate with the definitions of one of two semantically similar words. So, simply using such features cannot identify discriminative words precisely.

Finally, we combine both similarity and WordNet features together to address this challenge. There is no significant difference between CBOW based $v_c^{w1} v_c^{w2} v_t^{wA}$ and $v_t^{w1} v_c^{w2} v_c^{wA}$ in the validation set in terms of average F1. We select $v_c^{w1} v_c^{w2} v_t^{wA}$ as the winner combination of word and context embeddings, because this approach has closer F1 scores, when setting different la-

setup			positive=1			positive=0			average F1
			P	R	F1	P	R	F1	
validation	CBOW	$v_c^{w1}v_c^{w2}v_t^{wA}$	0.56	0.46	0.51	0.54	0.64	0.59	**0.55**
		$v_t^{w1}v_t^{w2}v_c^{wA}$	0.57	0.45	0.50	0.55	0.66	0.60	**0.55**
		$v_c^{w1}v_c^{w2}v_c^{wA}$	0.50	0.37	0.43	0.50	0.63	0.56	0.49
		$v_t^{w1}v_t^{w2}v_t^{wA}$	0.50	0.30	0.37	0.50	0.70	0.58	0.48
	Skip-gram	$v_c^{w1}v_c^{w2}v_t^{wA}$	0.54	0.38	0.45	0.52	0.68	0.59	0.52
		$v_t^{w1}v_t^{w2}v_c^{wA}$	0.55	0.47	0.51	0.54	0.62	0.58	**0.54**
		$v_c^{w1}v_c^{w2}v_c^{wA}$	0.52	0.33	0.40	0.51	0.70	0.59	0.49
		$v_t^{w1}v_t^{w2}v_t^{wA}$	0.53	0.41	0.46	0.52	0.64	0.57	0.52
test	CBOW	$v_c^{w1}v_c^{w2}v_t^{wA}$	0.54	0.56	0.55	0.63	0.62	0.63	**0.59**
		$v_t^{w1}v_t^{w2}v_c^{wA}$	0.54	0.53	0.53	0.62	0.63	0.63	0.58
		$v_c^{w1}v_c^{w2}v_c^{wA}$	0.50	0.47	0.49	0.59	0.62	0.61	0.55
		$v_t^{w1}v_t^{w2}v_t^{wA}$	0.49	0.39	0.44	0.58	0.67	0.62	0.53
	Skip-gram	$v_c^{w1}v_c^{w2}v_t^{wA}$	0.52	0.50	0.51	0.61	0.63	0.62	0.56
		$v_t^{w1}v_t^{w2}v_c^{wA}$	0.52	0.56	0.54	0.62	0.59	0.60	**0.57**
		$v_c^{w1}v_c^{w2}v_c^{wA}$	0.50	0.43	0.46	0.59	0.65	0.62	0.54
		$v_t^{w1}v_t^{w2}v_t^{wA}$	0.51	0.51	0.51	0.60	0.61	0.61	0.56

Table 2: Experimental results by using word and context embeddings (Feature 1-3).

setup		positive=1			positive=0			average F1
		P	R	F1	P	R	F1	
WordNet	validation	0.66	0.24	0.36	0.53	0.87	0.66	0.51
	test	0.66	0.26	0.37	0.60	0.89	0.71	0.54

Table 3: Experimental results by using WordNet definition features (Feature 4-5).

setup			positive=1			positive=0			average F1
			P	R	F1	P	R	F1	
CBOW+WordNet	validation	$v_c^{w1}v_c^{w2}v_t^{wA}$+WN	0.57	0.53	**0.55**	0.56	0.59	**0.57**	**0.56**
	test	$v_c^{w1}v_c^{w2}v_t^{wA}$+WN	0.58	0.60	**0.59**	0.66	0.65	0.65	**0.62**

Table 4: Final results by using CBOW word and context embeddings, and WordNet features (Feature 1-5).

bels (1 or 0) as the positive label. Thus, in the final submission, we use CBOW trained context embeddings of w_1 and w_2, and word embeddings of w_A to compute similarity features. We identify whether an attribute word w_A can distinguish w_1 and w_2 by using the above similarity features and WordNet definition features together. Although word and context embedding based similarity features are much more effective than WordNet features, by introducing WordNet features, the model further improves its performance, achieving 62% F1 on the test set (Table 4). WordNet definitions are also supportive features in this task.

Error analysis. We found that a significant portion of failures appear in those examples that the textual associations of the attribute words and the semantically similar words are not always discriminative. E.g., given a triple, (sons, father, young), our model failed in identifying *young* as a discriminative attribute, because *young* has been widely used to describe *sons* and *father* in the text (e.g., *young sons* and *a young father*). In this case, our word co-occurrence based method is suboptimal.

5 Conclusion

In this paper, we extended traditional word embedding methods (CBOW and Skip-gram) to distinguish two semantically similar words using an attribute word. In contrast with simply using traditional word representations, using both context and word embeddings can better model the co-occurrence between the two similar words and their discriminative attribute word. If the attribute word frequently co-occurs with one of the similar words more than another one within the same sentences, then the two semantically similar words can be distinguished by the attribute word. By using CBOW word and context embedding based similarity features and simple WordNet based word sense definition features, our model performs an average F1 of 62% on the test set.

Acknowledgments

This work is supported by the award made by the UK Engineering and Physical Sciences Research Council (Grant number: EP/P005810/1).

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of International Conference on Learning Representations*.

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 83–84. International World Wide Web Conferences Steering Committee.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Xin Rong. 2014. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.