THU_NGN at SemEval-2018 Task 10: Capturing Discriminative Attributes with MLP-CNN model

Chuhan Wu[1], **Fangzhao Wu**[2], **Zhigang Yuan**[1], **Sixing Wu**[1] **and Yongfeng Huang**[1]
[1]Tsinghua National Laboratory for Information Science and Technology,
Department of Electronic Engineering, Tsinghua University Beijing 100084, China
[2]Microsoft Research Asia
{wuch15,yuanzg14,wu-sx15,yfhuang}@mails.tsinghua.edu.cn
wufangzhao@gmail.com

Abstract

Existing semantic models are capable of identifying the semantic similarity of words. However, it's hard for these models to discriminate between a word and another similar word. Thus, the aim of SemEval-2018 Task 10 is to predict whether a word is a discriminative attribute between two concepts. In this task, we apply a multilayer perceptron (MLP)-convolutional neural network (CNN) model to identify whether an attribute is discriminative. The CNNs are used to extract low-level features from the inputs. The MLP takes both the flatten CNN maps and inputs to predict the labels. The evaluation F-score of our system on the test set is 0.629 (ranked 15th), which indicates that our system still needs to be improved. However, the behaviours of our system in our experiments provide useful information, which can help to improve the collective understanding of this novel task.

1 Introduction

Evaluating the similarity of words is an important task in semantic modeling. There have been different approaches based on corpus statistics (Jiang and Conrath, 1997; Mihalcea et al., 2006) and ontology (Seco et al., 2004; Sánchez et al., 2012). After an effective word representation proposed by mikolov et al (2013), word similarity can be evaluated based on word embedding weights (Levy and Goldberg, 2014). Usually higher cosine similarity of word embedding vectors indicates higher semantic similarity.

However, existing semantic methods are not capable of discriminating similar words between each other without additional information. For example, it is easy for these models to tell "dog" and "puppy" is similar, but they can't tell the differences between each other. It limits the use of these models to mine such fine-grained semantic information from texts. Thus, the SemEval-2018 Task 10 is proposed to determine whether an attribute can help to discriminate between two words(Krebs et al., 2018). One can express semantic differences between concepts by referring to attributes associated with those concepts. The differences between concepts can usually be identified by the presence or absence of specific attributes. For example, the attributes "red" and "yellow" are discriminative for concepts "apple" and "banana", while "sweet" or "fruit" are not discriminative.

Capturing such discriminative attributes can be regarded as a binary classification task: given two words and an attribute, predict whether the attribute is a difference between the two words. Existing methods to capture discriminative attributes are mainly based on dictionary (Parikh and Grauman, 2011). In recent years, CNN have been successfully applied to text classification task (Kim, 2014). In order to address this task, we develop a system based on MLP-CNN model. Firstly, the input words will be converted into dense vectors using the combination of different word embeddings. Then the CNN layers are used to extract features from these vectors. Finally, a MLP classifier is used to predict binary labels based on both embedding and CNN features. The experimental results show that our model outperforms several baseline neural model, and the additional features can improve the model performance. Our system still has room for development according to the experimental analysis. The behaviours of our system in our experiments can help to further fix and extend our model.

2 MLP-CNN model with Word Feature

2.1 Network Architecture

The architecture of our MLP-CNN model is shown in Figure 1. The input of our system is a pair of words with an attribute. First, an embedding layer is used to provide different kinds of pre-trained embedding weights ($v_1 - dim$) and the word features ($v_f - dim$) . We use three different pre-trained embedding weights and concatenate them together with the additional features of each word. Thus, the output of embedding layer is $(v_1 + v_f) - dim$.

Second, a 2-layer convolutional neural network take these vectors as input, and output the flatten feature maps. We use zeros to pad in both sides to keep the same output length. Since the length of inputs is 3, the 3 time steps of the convolutional feature maps can respectively extract the inherent relatedness of the first word with attribute, the second word with attribute and all three words. The feature map dimensions of the two CNN layers are v_2 and v_3 respectively. In order to reduce the difficulties of gradient propagation in neural networks, we use a over-layer connection between input and output of CNN. We concatenate the flatten feature maps with all word embedding and features together. Finally a MLP with ReLU and sigmoid activation function is used to predict the normalized binary label. With the help of the over-layer connection, the MLP classifier can learn from high-level word information and raw semantic information at the same time. Since the final labels are obtained from the triples of words through the embedding, CNN and dense layers, all parameters can be tuned in model training.

2.2 Word Embedding

Since there are several out-of-vocabulary words in the dataset when using single pre-trained word embedding, we use three different embedding models to cover them. The three embedding models include pre-trained word2vec embedding[1] provided by Mikolov et al. (Mikolov et al., 2013), the Glove embedding[2] provided by Pennington et al. (Pennington et al., 2014) and the fastText embedding[3] released by bojanowski et al. (Bojanowski et al., 2016). These embedding weights are all 300-dim. They are concatenated together as the representation of input words.

2.3 Word Feature

In our model, we use one-hot encoded POS tags and two binary features obtained by Word-

[1]https://code.google.com/archive/p/word2vec/

[2]http://nlp.stanford.edu/data/glove.840B.300d.zip

[3]https://s3-us-west-1.amazonaws.com/fasttext-vectors/wiki.en.vec

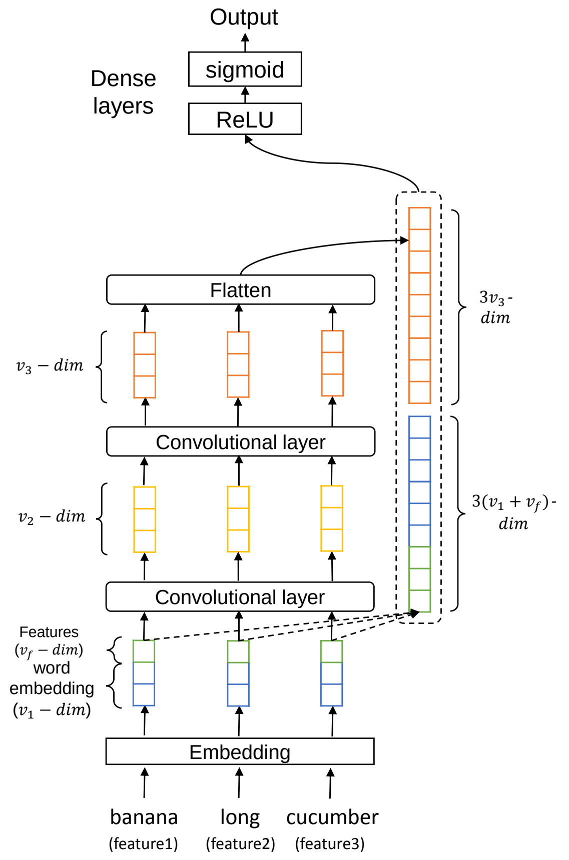

Figure 1: The architecture of our MLP-CNN model.

Net (Miller and Fellbaum, 1998). In the dataset, the words to be discriminated are nouns, but attributes are nouns, adjectives, verbs and so on. Thus, POS tags of words can help to identify the types of attributes and the relationship between them.)We use the Stanford parser tool[4] to get the POS tags of words.

The WordNet feature we use is based on synsets. Among every three input words, if one word is in the synset of another word, the corresponding feature digit will be set to 1, or it will be set to 0. In this way, a 2-dim synset feature of each word can be obtained. We use the nltk tool(Bird et al., 2009) to generate the WordNet features. The features above are concatenated with word embedding as the input of MLP-CNN model.

2.4 Model Training and Ensemble

Since the *train set* is unbalanced, we randomly select same numbers of positive (the attribute is discriminative) and negative (the attribute is not discriminative) samples from the *train set* every time. Thus, the training data we used in our experiments

[4]https://nlp.stanford.edu/software/lex-parser.shtml

consists of the sampled data from the *train set* and 80% data sampled from the *dev set*. The remaining 20% part in *dev set* is used for validation.

Model ensemble strategy has been proven useful to neural networks (Wu et al., 2017). Therefore, we build different training samples using the method described in the above paragraph and train our model for 10 times. The final predictions on the test set are the average of the predictions of the 10 models. In this way, the performance of neural model can be further improved.

3 Experiment

3.1 Experiment Settings

The dataset we use is constructed based on the approach proposed by Lazaridou et al. (2016) and the initial source of data is provided by McRae et al. (2005). The entire dataset contains 17,547 samples for training, 2,722 for validation and 2,340 for testing. The training set is automatically generated, while the validation and test set are manually refined. The models will be evaluated by F1-measure, as is standard in a binary classification task.

In our network, the kernel sizes of CNN are set to 3. The dimensions of feature maps v_2 and v_3 are set to 256, and the dimensions of dense layers are 300. The dropout rate of both embedding weights and CNN is set to 0.2. The training batch size is set to 50. We use Adam as the optimizer for network training, which takes 10 epochs per time. We train our model for 10 times and average their predictions on the test set.

3.2 Performance Evaluation

The experimental results on the test and validation set are shown in Table 1. For comparison, we also present several baseline models here. Our official submission is the MLP-CNN model with ensemble techniques. Our F-score is 62.9 (ranked 15th) in the evaluation phase. From the evaluation results, we can see that our model outperforms these baseline models. It shows that our network architecture can learn more semantic information from the words and attributes. However, our system needs to be improved compared with the top system (75.0 of F-score). In addition, the testing results are much lower than validation results. Some detailed information will be analysis in the next section.

Model	**F-score**	
	test	validation
MLP	53.5	63
CNN	57.8	66.2
MLP-CNN	61.7	71.5
MLP-CNN w/o over-layer	61.0	70.3
MLP-CNN+ensemble (ours)	**62.9**	**73.4**

Table 1: Performance evaluation of our system and several baselines.

4 Discussion

4.1 Influence of Trainable Word Embedding

The influence of different word embedding weights and fine-tuning them or not is shown in Table 2. Note that we don't apply the model ensemble technique here. From the results, we can see that the combination of different word embedding can significantly improve the model performance. It may be because that using different word embedding can provide richer semantic information. In addition, using the combinations of different word embedding can cover more words and the out-of-vocabulary words in the single embedding file can be reduced. Thus, the predictions of such words can be more accurate.

However, we find fine-tuning the pre-trained word embedding is not a good choice. The fine-tuned model performance is significantly worse than models using untrainable embedding. Since the training, validation and test sets have no feature overlap between them, fine-tuning the embedding weights will lead to serious over-fitting and poor model generalization ability. We fine-tuned the embedding of our models used in the official submission, so the results are lower than the models with untrainable embedding.

4.2 Influence of Word Features

The influence of the two types of features is shown in Table 3. The results show that additional word features can improve the performance of our neural model. Attributes with different POS tags provide different semantic information. For example, given a pair of words "boy" and "woman", the attributes "young" and "run" describe very different aspects. Therefore, POS tag features can help the model to extract different features from the input words. Another feature based on WordNet can also improve our model. It may be because if the attribute is in the synsets of a concept, it's usu-

pre-trained Embedding	**F-score**	
	test	validation
w/o	44.9	63.7
word2vec+tune	50.6	65.8
Glove+tune	53.5	66.7
fastText+tune	51.5	64.4
all+tune	61.7	71.5
word2vec-tune	54.9	66
Glove-tune	57.7	69.6
fastText-tune	58.3	68.9
all-tune	**65.7**	**75.5**

Table 2: Comparisons of using different pre-trained embedding.

ally an attribute of this concept. Thus, the synset information can help the model to identify the relationships between words and attibutes.

Features	**F-score**
w/o	59.2
+POS	61
+WordNet	60.1
+POS+WordNet	**61.7**

Table 3: Influence of word features on the test set.

4.3 Case Study

Several examples of model predictions on the test set are shown in Table 4. From the true predictions, we can see that our model can capture simple attributes of concepts such as colors. However, more complex relationships between words and attributes are difficult for our system to mine. For example, the word "mouse" can be an animal or electronic device. It's hard to identify such semantic differences without incorporating external knowledge, since the information provided by the training data is limited.

True Positive	**False Positive**
corn,tomato,yellow	alcohol,liquor,strong
ant,snail,black	bar,shop,sell
True Negative	**False Negative**
father,brother,family	mouse,dog,plastic
father,mother,parent	engine,vehicle,component

Table 4: Representative examples of the predictions on the test set.

5 Conclusion

Discriminating similar words between each other without additional information is difficult for existing semantic models. Therefore, the SemEval-2018 task 10 is proposed to fill this gap. In this paper, we apply a MLP-CNN model with word feature to this task. In our model, the input and output of our CNN are highway connected. They are taken by a MLP classifier for binary classification. Based on this model, the local relationships between each pair of words can be mined. Our evaluation F-score is 62.9 (ranked 15th). The detailed analysis on our system shows our system can be further improved.

Acknowledgments

The authors thank the reviewers for their insightful comments and constructive suggestions on improving this work. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB0800402 and in part by the National Natural Science Foundation of China under Grant U1705261, Grant U1536207, Grant U1536201 and U1636113.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606.*

Jay J Jiang and David W Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008.*

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882.*

Alicia Krebs, Alessandro Lenci, and Denis Paperno. 2018. Semeval-2018 task 10: Capturing discriminative attributes. In *Proceedings of the 12th international workshop on semantic evaluation (SemEval 2018).*

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2016. The red one!: On learning to refer to things based on their discriminative properties. *arXiv preprint arXiv:1603.02618.*

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

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.

Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

George Miller and Christiane Fellbaum. 1998. Wordnet: An electronic lexical database.

Devi Parikh and Kristen Grauman. 2011. Interactively building a discriminative vocabulary of nameable attributes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1681–1688. IEEE.

Jeffrey Pennington, Richard Socher, and Christopher 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.

David Sánchez, Montserrat Batet, David Isern, and Aida Valls. 2012. Ontology-based semantic similarity: A new feature-based approach. *Expert Systems with Applications*, 39(9):7718–7728.

Nuno Seco, Tony Veale, and Jer Hayes. 2004. An intrinsic information content metric for semantic similarity in wordnet. In *ECAI*, volume 16, page 1089.

Chuhan Wu, Fangzhao Wu, Yongfeng Huang, Sixing Wu, and Zhigang Yuan. 2017. Thu_ngn at ijcnlp-2017 task 2: Dimensional sentiment analysis for chinese phrases with deep lstm. *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 47–52.