pkudblab at SemEval-2016 Task 6 : A Specific Convolutional Neural Network System for Effective Stance Detection

Wan Wei, Xiao Zhang, Xuqin Liu, Wei Chen, Tengjiao Wang
Key Laboratory of High Confidence Software Technologies, Ministry of Education,
School of Electronics Engineering and Computer Science,
Peking University, Beijing 100871, China
{wanw, zxcs, xuqinliu, pekingchenwei, tjwang}@pku.edu.cn

Abstract

In this paper, we develop a convolutional neural network for stance detection in tweets. According to the official results, our system ranks 1^{st} on subtask B (among 9 teams) and ranks 2^{nd} on subtask A (among 19 teams) on the twitter test set of SemEval2016 Task 6. The main contribution of our work is as follows. We design a "vote scheme" for prediction instead of predicting when the accuracy of validation set reaches its maximum. Besides, we make some improvement on the specific subtasks. For subtask A, we separate datasets into five sub-datasets according to their targets, and train and test five separate models. For subtask B, we establish a two-class training dataset from the official domain corpus, and then modify the softmax layer to perform three-class classification. Our system can be easily re-implemented and optimized for other related tasks.

1 Introduction

There are several requirements for stance detecting applications on the internet. However it is unpractical for humans to classify massive amounts of tweets. Twitter stance detection aims to automatically determine the emotional tendency of tweets. To classify tweets polarity, mainstream approaches are based on Pang (Pang et al., 2002), like regression problem, using machine learning algorithm to build classifiers from tweets with manually annotated polarity to classify the polarity of a tweet(Jiang et al., 2011; Hu et al., 2013; Dong et al., 2014). In this direction, most studies focus on designing effective features to obtain better classification performance (Pang and Lee, 2008; Liu, 2012; Murakami and Raymond, 2010). For example, Mohammad (Mohammad and Turney, 2013) implements some sentiment lexicons and several manually-selected features. To leverage massive tweets containing positive and negative emoticons for automatically feature learning, Tang (Tang et al., 2014) proposes to learn sentiment-specific word embedding. We transfer this method to detect tweets stance.

In this paper, we develop a specific convolutional neural network learning model for stance detection. Firstly, we learn word embedding from Google News database as the input of our system. Afterwards, we train the CNN model with the SemEval2016 Task 6 dataset. Finally, we design a "vote scheme" using the softmax results to predict the label of test set. We also make some task specific improvement. For subtask A, we separate datasets into five sub-dataset, and train and test five separate models. For subtask B, we establish a two-class training dataset from the official domain corpus based on several special expressions. We evaluate our deep learning system on the test set of SemEval2016 Task 6. Our system ranks 1^{st} on subtask B and 2^{nd} on subtask A. The good performance in the Task 6 evaluation verifies the effectiveness of our model and schemes.

2 Architecture overview

The architecture of our convolutional neural network is mainly inspired by the architecture proposed by Kim, which performs well and efficiently in sentence classification tasks (Kim, 2014). The reason

Proceedings of SemEval-2016, pages 384–388,
San Diego, California, June 16-17, 2016.

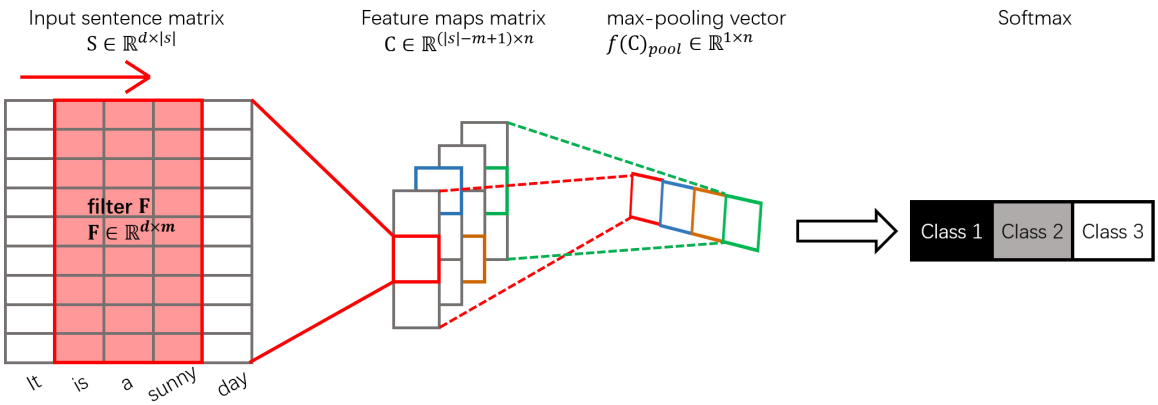

Figure 1: Main archtecture of our convolutional neural network.

why we base on Kim's model is that there is much in common between stance detection task and sentence classification task when the amount and the distribution of dataset is rather reasonable. Our architecture is shown on Fig. 1.

In the following, we give a brief introduction of the main components of our network architecture in the connecting order: look-up table, input matrix, convolutional layer, activation function, pooling layer and softmax layer. We also describe the approach to train this model.

2.1 Look-up table

Look-up table is a huge word embedding matrix. Each column of the table, which is d-dimensional, corresponds to a word. Word embedding in the look-up table are pre-trained vectors published by word2vec team (Mikolov et al., 2013)[1]. These vectors are trained on part of Google News dataset (about 100 billion words).

2.2 Input matrix

An input matrix S, $\mathrm{S} \in \mathbb{R}^{\mathrm{d} \times |\mathrm{s}|}$, is the representation of an input sentence: $[\mathrm{w}_1, \mathrm{w}_2, ..., \mathrm{w}_{|\mathrm{s}|}]$. $|\mathrm{s}|$ is the length of the sentence, w_i is the corresponding d-dimensional vector found in look-up table. If this word does not exist in the look-up table, make it a zero vector or a vector whose components are numbers randomly generated in a given range.

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

2.3 Convolutional layer

The goal of the convolutional layer is to extract patterns, so that some common abstractive representation can be found among the dataset. Pattern means specific sequential words in a sentence. Patterns can be extracted by different filter matrixes F which are discriminatively sensitive to different patterns.

More formally, the convolution operation $\odot$ between an input sentence matrix $\mathrm{S} \in \mathbb{R}^{\mathrm{d} \times |\mathrm{s}|}$ and a filter $\mathrm{F} \in \mathbb{R}^{\mathrm{d} \times \mathrm{m}}$, where m is an assigned width, is defined as follow:

$$\mathrm{c_i} = \sum (\mathrm{S}_{[:,\mathrm{i}:\mathrm{i}+\mathrm{m}-1]} \odot \mathrm{F}),$$

where $1 \leq \mathrm{i} \leq |\mathrm{s}| - \mathrm{m} + 1$. $\mathrm{S}_{[:,\mathrm{i}:\mathrm{i}+\mathrm{m}-1]}$ is a matrix slice of size m along the columns and $\odot$ is the element-wise multiplication. Both S and F have the same d rows. As shown on Fig. 1, filter F slides along the column dimension of S generating vector c: $[\mathrm{c}_1, \mathrm{c}_2, ..., \mathrm{c}_{|\mathrm{s}|-\mathrm{m}+1}]$, named feature map.

So far we have introduced how to compute a convolution between the input sentence matrix and a single filter. To get a richer representation of the dataset, we apply n filters on every input sentence matrix to compute feature maps matrix C, $\mathrm{C} \in \mathbb{R}^{(|\mathrm{s}|-\mathrm{m}+1) \times \mathrm{n}}$. Note that every input sentence matrix has a corresponding C matrix and every column of matrix C corresponds to a convolution result between a filter and this input sentence matrix.

In practice, we also add a bias vector $\mathrm{b} \in \mathbb{R}^n$ to every row of matrix C element-wise to train a more appropriate model.

2.4 Activation function

To fit the non-linear boundaries better, convolutional layer is always followed by a non-linear activation function $f()$ in practice. $f()$ is applied element-wise on feature maps matrix C. Among the most popular choices of activation functions: sigmoid, tanh (hyperbolic tangent) and ReLU (rectified linear), we finally choose ReLU, since it is rather simple and sometimes more efficient[2].

2.5 Pooling layer

For the purpose of simplifying the information in the output from the convolutional layer (passed through the activation function), pooling layer is used. We adopt the max-pooling method, which is choosing the maximum value from every column of $f(C)$ ($f()$ is the ReLU operation), to form a condensed representation vector. More formally, after the max-pooling operation, $f(C) \in \mathbb{R}^{(|\mathrm{s}|-\mathrm{m}+1)\times \mathrm{n}} \rightarrow pool(f(C)) \in \mathbb{R}^{1\times \mathrm{n}}$, which is also shown on Fig. 1.

2.6 Output layer: softmax

The fully connected softmax layer is for classification. To a K-class dataset, the probability distribution of j-th class is as follows:

$$\mathrm{P}(\mathrm{y}=\mathrm{j}|\mathrm{x},\mathrm{s},\mathrm{b}) = \mathrm{softmax_j}(\mathrm{x^T w}+\mathrm{b})$$

$$= \frac{\mathrm{e}^{\mathrm{x^T w_j}+\mathrm{b_j}}}{\sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{e}^{\mathrm{x^T w_k}+\mathrm{b_k}}},$$

where x is the input vector (the vector produced by pooling layer in our network), $\mathrm{w_k}$ and $\mathrm{b_k}$, having the same dimensionality as the input vector x, are weight vector and bias vector of the k-th class respectively.

Softmax layer calculates the probability of each class and then chooses the class having the maximum value as the predict label.

2.7 Approach to train the network

The parameters trained by our network are as follows:

$$\theta = \{\mathrm{W}; \mathrm{F}; \mathrm{b}; \mathrm{w_k}; \mathrm{b_k}\},$$

where W is the word embedding of all words in dataset, including those found in the look-up table and those randomly assigned; F is the set of all the filters; b is the bias vector in the convolutional layer; $\mathrm{w_k}$ and $\mathrm{b_k}$ are the weight and the bias vector of k-th class in the softmax layer.

We use backpropogation algorithm to optimize these parameters and we adopt Adadelta (Zeiler, 2012) update rule to automatically tune the learning rate. We also opt our network by another two methods: l2-norm regularization terms for the parameters to mitigate overfitting issues and dropout scheme (Srivastava et al., 2014), which is to set the chosen value zero, to prevent feature co-adaptation.

3 Improvement for stance detection

In this session, we briefly introduce our improvement on the CNN archtecture we described above. The improvement is task specific.

Vote scheme. We validate our model by cross validation method. For the models of subtask A and subtask B, we design ten parallel epochs, whose validation sets are randomly selected from the training set and non-overlap.

Different from general network, we design a "vote scheme" for prediction instead of predicting when the accuracy of validation set reaches its maximum. In each epoch, we choose some iterations deliberately to predict the test set. Then, when this epoch ends, for every sentence in the test set, we appoint the label which appears most frequently in these predictions as the result of this epoch. Finally, when ten epochs end, we vote within results of these ten epochs by the same method described above to determine the final labels.

By performing multiple times independently and voting twice, we get a rather robust mechanism for predicting.

"divide and conquer" scheme. For subtask A, we separate both training and test datasets respectively into five sub-datasets according to their targets, and then train and test five separate models with these divided datasets. The contrast experiment between this "divide and conquer" model and the model trained by the integral dataset is shown in Session 4.

"2-step" scheme. For subtask B, in the condition

[2]Rough experiment have been done on the training set with different activation functions, since these experiment are not that thoughtful, the result will not be shown in experiment session.

Corpus	Favor	Against	None	Total
Training Dataset				
Subtask A	753	1,394	766	2,913
Subtask A-Atheism	92	304	117	513
Subtask A-CC[1]	212	15	168	395
Subtask A-FM[2]	210	328	126	664
Subtask A-HC[3]	118	393	178	689
Subtask A-LofA[4]	121	354	177	652
Subtask B	2,562	3,418	—	5,980
Test Dataset				
Subtask A	304	715	230	1,249
Subtask A-Atheism	32	160	28	220
Subtask A-CC[1]	123	11	35	169
Subtask A-FM[2]	58	183	44	285
Subtask A-HC[3]	45	172	78	295
Subtask A-LofA[4]	46	189	45	280
Subtask B	148	299	260	707

[1] Target "Climate Change is a Real Concern".
[2] Target "Feminist Movement".
[3] Target "Hillary Clinton".
[4] Target "Legalization of Abortion".

Table 1: Task 6 dataset.

Corpus	"Divide"$_1$ Model	"Integral"$_1$ Model
subtask A-all	0.6733	0.6498
subtask A-Atheism[2]	0.6334	0.5652
subtask A-CC[2]	0.5269	0.5379
subtask A-FM[2]	0.5133	0.5377
subtask A-HC[2]	0.6441	0.6172
subtask A-LofA[2]	0.6109	0.6098

[1] Official metric: (F_favor + F_against) / 2.
[2] Abbreviation of targets. Those have been described in the footnote of Table 1.

Table 2: Contrast experiment results on subtask A. "Divide" refers that those results come from five separate model trained by five target corpus. "Integral" refers that those results come from one model trained by integral subtask A corpus.

that the official corpus is unlabeled whereas training set is necessary for our supervised model, we come up with a solution having two steps: 1. Build a two-class training dataset; 2. Modify the softmax layer to perform three-class classification on the two-class training dataset.

According to some expressions and hashtags revealing a distinct tendency, for example, "go trump" and "#MakeAmericaGreatAgain" reveal favor tendency whereas "idiot" and "fired" reveal against tendency, we finally establish a two-class training dataset, which has about 2000 favor tweets and about 3000 against tweets, from the domain corpus for subtask B (Mohammad et al., 2016). Then, we modify the softmax layer. For a test sentence, if the absolute value of the subtraction between the probability values of the two classes is less than a randomly selected real number $\alpha(\alpha \in [0.05, 0.1])$, predict this sentence as "None" stance. Otherwise, predict it the class having the greater probability value.

4 Experiments and evaluation

Dataset. For subtask A, the training set is the official training data for Task A (Mohammad et al., 2016). For subtask B, the training set is described in Session 3. Details about datasets are shown in Table 1.

Parameters setup. Word embedding matrix is described in Session 2.1, the dimensionality d is 300. We design three different width filters, 100 in width 3, 100 in width 4 and 100 in width 5, which means that there are 300 filters in total. We choose ReLU as activation function and we use max-pooling. L2-norm regularization term is set to 1e-6, the probability of dropout is set to 0.5. Bias vector b, as well as $\mathrm{w_k}$ and $\mathrm{b_k}$ in softmax layer are all set to zero vectors.

Test result. We perform contrast experiments on subtask A. The results of the "divide and conquer" model and its contrast integral model, as well as the five separate models are shown in Table 2. The description of these models is in Session 3. We can see from Table 2 that "divide and conquer" model does not always have a better performance. However, since the words using in the sentences which belong to the same target are expected to be more similar, the "divide" model still performs much better on some dataset (e.g. Atheism). The "divide and conquer" model is the one we submit for evaluation.

Official ranking. Part of the official rankings for both subtask A and B are summarized in Table 3. As we can see our model performs well on both subtasks. Our model ranks 2^{nd} on subtask A, whose official metric is only 0.5% lower than the first team. On subtask B our model ranks 1^{st}, and the official metric is 56.28%, about 10% higher than the second team.

Team	Official Metric[1]	Rank
Subtask A		
MITRE	0.6782	1
pkudblab	0.6733	2
TakeLab	0.6683	3
PKULCWM	0.6576	4
#Total teams: 19		
Subtask B		
pkudblab	0.5628	1
LitisMind	0.4466	2
INF-UFRGS -OPINION-MINING	0.4232	3
#Total teams: 9		

[1] Official metric: (F_favor + F_against) / 2.

Table 3: Part of the official result.

5 Conclusions

We develop a specific convolutional neural network system for detecting twitter stance in this paper. We give a detailed description of our model and specific adaptation for different subtasks. Among 28 submitted systems, our system obtains good rank on both subtask A and subtask B on the test set of SemEval2016 Task 6. Our system has good scalability for other related tasks.

6 Future work

Due to the tight schedule, there are still many aspects need to explore. For example, why the Google news word2vec performs well in this context? How much does this word embedding improve the score compared with randomly initial word embedding? Is the more suitble word embedding exists? What's more, the vote scheme is somewhat curt, we should do more experiment to validate its robustness. Our code is available in github for anyone who has a interest in further exploration[3].

References

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *ACL (2)*, pages 49–54.

Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. 2013. Unsupervised sentiment analysis with emotional signals. In *Proceedings of the 22nd international conference on World Wide Web*, pages 607–618. International World Wide Web Conferences Steering Committee.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 151–160. Association for Computational Linguistics.

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

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

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

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Saif M. Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, June.

Akiko Murakami and Rudy Raymond. 2010. Support or oppose?: classifying positions in online debates from reply activities and opinion expressions. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 869–875. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)*, pages 1555–1565.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

[3] https://github.com/nestle1993/SE16-Task6-Stance-Detection