Neural Classifier with Statistic Information of User and Product for Sentiment Analysis

Changliang Li[1], Jiayu Xie[2], and Yaoguang Xing[3]

[1] Kingsoft AI Lab
lichangliang@kingsoft.com
[2] University College London, London
ucakjxi@ucl.ac.uk
[3] Tsinghua University, Beijing
xyg17@mails.tsinghua.edu.cn

Abstract. Sentiment analysis models based on neural network architecture have achieved promising results. Some works bring improvement to these neural models via taking user and product into account. However, the way of utilizing significant role user and product by now is limited to embed them into vectors on word or semantic level, and ignore statistic information carried by them such as all the marks given by one user. In this paper, we propose a novel neural classifier, which extracts and feeds statistic information carried by user and product to neural networks. Our proposed method can utilize user preference and product characteristics so as to yield excellent performance on sentiment analysis. To fully evaluate the efficiency of our model, we conduct experiment on three popular sentiment datasets: IMDB, Yelp13 and Yelp14. And the experiment results show that our model achieves state-of-the-art on all three datasets.

Keywords: Sentiment Analysis · Neural Classifier · Natural Language Processing.

1 Introduction

Sentiment analysis, also called opinion mining, is the field of study that analyzes people's opinion and sentiments towards entities such as products, events and topics and so on [1]. In general, sentiment analysis has been investigated mainly at three levels: document level, sentence level and entity and aspect level [1]. Our work focuses on document level. Sentiment analysis at document level is to distinguish humans' opinion and sentiment that a document expresses.

To involve statistic information into sentiment analysis, we firstly feed the statistic representation of the user and product to two independent neural networks with same architecture. Meanwhile, we use Long Short-Term Memory network to generate semantic representation of document of pure text. And then we merge the representation of statistic and text information in vector space and

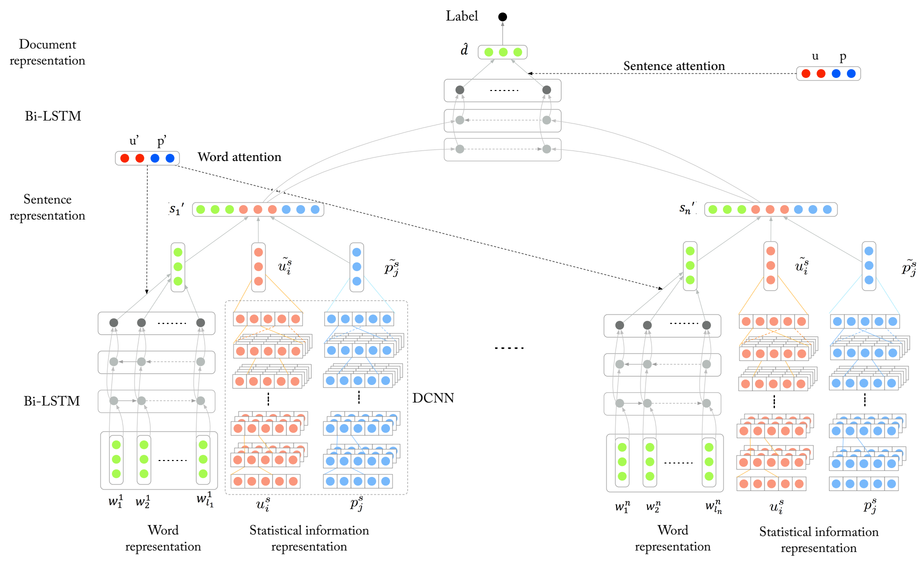

Fig. 1. Neural Classifier with Statistic Information

predict the sentiment label depend on it. The experiment results show that our model achieves state-of-the-art on all three datasets.

2 Related Work

In recent years, models based on CNN for sentiment classification have achieved great success[2]. [2] utilized CNN to classify sentence class with word level embedding, which constructs words with their embedding as a matrix and applied with convolution and max pooling. Recurrent neural network is another main approach for sentiment classification because it can capture the sequential information. [3, 4] involve global user and product information into Long Short-term Memory network. Besides, a series of recursive neural network models such as Recursive Neural Tensor Network (RNTN) [6] were proposed for sentiment analysis and gave good performance. Moreover, attention has become an effective mechanism to obtain superior results[7].

Despite the progress of those methods have achieved, it is still a challenging and open task for sentiment analysis. Therefore, we are motivated to design a powerful model, which can fully employ statistic information for sentiment classification.

3 Neural Classifier with Statistic Information

In this section, we give the description of our neural classifier with statistic information in detail. Figure 1 gives the overall architecture of our model. We employ deep convolutional neural networks (DCNN) for representing statistic information of user and product, and bidirectional long short-term memory (Bi-LSTM) network for pure text. Attention on word and sentence level is involved to compute sentence and document representation in vector space. Document representation in vector space is used as features for sentiment classification.

3.1 Statistic Information of User and Product

In our work, we build a user-product score matrix $M_{UP} \in \mathbb{R}^{n_u \times n_p}$, where n_u represents the number of users and n_p represents the number of products. The element m_{ij} in matrix represents the score of the j-th product marked by the i-th user. If the i-th user did not rate the j-th product, we set $m_{ij} = -1$. Because each user rates only a small amount of product in the corpus, the user-product score matrix M_{UP} is very sparse given big data.

To address this problem, we extract the principle component features of M_{UP} via principal component analysis (PCA) algorithm. We obtain a new matrix $M_U \in \mathbb{R}^{n_u \times d_p}$ via applying PCA over user rows, where d_p means the reduced dimension. Each row of this matrix u_r^s represents the statistic information of the r-th user.

In the similar way, we obtain another matrix $M_P \in \mathbb{R}^{n_p \times d_u}$ via applying PCA over the product rows of M_{UP}^T where T means transpose, and d_u means the reduced dimension. Each row of this matrix p_r^s represents the statistic information of the r-th product.

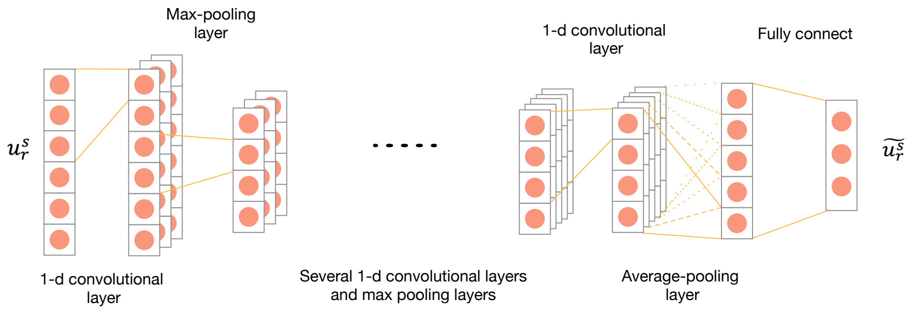

Fig. 2. DCNN architecture for statistic information

3.2 Hidden Representation for Statistic Information

Based on representations u_r^s and p_r^s, we utilize DCNN architecture to capture hidden representation $\widetilde{u_r^s}$ and $\widetilde{p_r^s}$ of statistic information respectively. The details are given as follow.

Because the user and product shares the same computation process, we take user as illustration in this section. Fig.2 shows the DCNN architecture employed to extract statistic information of user.

The DCNN architecture consists of multiple 1-dimensional convolutional layers, max-pooling layers and a global average-pooling layer.

Given an input vector u_r^s, there is a discrete input function $g(x, u_r^s) : \{1, 2, ..., l\} \to \mathbb{R}$, which extracts the x-th value of the vector u_r^s, and a discrete kernel function $f(x) : \{1, 2, ..., k\} \to \mathbb{R}$, where l represents the vector length of statistic vector u_r^s, and k represents the kernel size. The employed 1-dimensional convolution operation $h(t, u_r^s)$ with stride d_c and kernel function $f(x)$ is defined as the following equation:

$$h(t, u_r^s) = b + \sum_{a=1}^{k} f(a) \cdot g(t \cdot d_c + a, u_r^s) \tag{1}$$

Where a and $t \cdot d_c + a$ are the indices of the kernel and input respectively, b is a bias term and $d_c = 1$. The feature map u_r^c obtained by activation function ReLU [8] and above convolution operation is defined as:

$$u_r^c = [ReLU(h(1, u_r^s)), ..., ReLU(h(l-k+1, u_r^s))] \tag{2}$$

Afterwards, we utilize max-pooling layer to extract features. The max-pooling operation $m(t, u_r^c)$ with stride d_m is defined as:

$$m(t, u_r^c) = \max_{a=1}^{k} g(t \cdot d_m + a, u_r^c) \tag{3}$$

Then we obtain the output vector of this layer as follow:

$$u_r^m = [m(0, u_r^c), m(1, u_r^c), ..., m(l_m, u_r^c)] \tag{4}$$

Where $l_m = \left\lfloor \frac{(l-k+1)-k}{d_m} \right\rfloor$. Then the output of max-pooling layer u_r^m is followed by another 1-dimensional convolutional layer and max-pooling layer. The rest processes are made in the same manner until the last layer. The last convolutional layer is followed by a global average-pooling layer rather than max-pooling layer. The global average pooling avg with the output $\hat{u_r^c}$ of the last convolutional layer is defined as:

$$\overline{u_r^c} = avg(\hat{u_r^c}) = \frac{\sum_{a=1}^{L} g(a, \hat{u_r^c})}{L} \tag{5}$$

Where L is the length of vector $\hat{u_r^c}$. In our description above, one feature $\overline{u_r^c}$ is obtained by one kernel. And the DCNN uses multiple kernels, so we obtain multiple features which are concatenated together to obtain vector v. The vector v is followed by a fully connected layer:

$$\widetilde{u_r^s} = tanh(W \cdot v + b) \tag{6}$$

We regard the output of the fully connected layer $\widetilde{u_r^s}$ as the representation of the user statistic information. Similarly, the representation of product statistic information $\widetilde{p_r^s}$ is obtained in the same way.

3.3 Representation for Pure Text

Given a sentence of pure text, which consists of a sequence of words $\{w_1^i, w_2^i, ..., w_{l_i}^i\}$, we firstly represent all words using their corresponding embeddings $\{e_1^i, e_2^i, ..., e_{l_i}^i\}$, where e_j^i means word embeddings of the j-th word. We then employ Bi-LSTM network to obtain sentence representation in vector space.

Firstly, we compute the forward output $\overrightarrow{h_t^i}$ and backward output $\overleftarrow{h_t^i}$ by the following equations.

$$\overrightarrow{h_t^i} = \mathcal{H}(e_j^i, \overrightarrow{h_{t-1}^i}),\ \overleftarrow{h_t^i} = \mathcal{H}(e_j^i, \overleftarrow{h_{t-1}^i}) \tag{7}$$

Where $\mathcal{H}$ denotes the recurrent unit of Bi-LSTM ; $\overrightarrow{h_{t-1}^i}$ means previous hidden state;

As a result, we can obtain the final h_t^i as follow:

$$h_t^i = \overrightarrow{h_t^i} + \overleftarrow{h_t^i} \tag{8}$$

Moreover, in sentiment analysis task, every word from different users and products contributes differently. So we use attention mechanism to capture this information. Since we have a series of hidden state sequence $\{h_1^i, h_2^i, ..., h_n^i\}$ for a sentence, then we compute its hidden state s_i by the following equation:

$$s_i = \sum_{j=1}^{l_i} \alpha_j^i h_j^i \tag{9}$$

α_j^i means word level attention weight and is obtained by:

$$\alpha_j^i = \frac{\exp(\tau(h_j^i, \mathrm{u}, \mathrm{p}))}{\sum_{k=1}^{l_i} \exp(\tau(h_k^i, \mathrm{u}, \mathrm{p}))} \tag{10}$$

We embed each user and each product into vectors u and p. where τ is a function to calculate the importance of the word. τ is defined as:

$$\tau(h_k^i, \mathrm{u}, \mathrm{p}) = v^{\mathrm{T}} \tanh\left(W_H h_k^i + W_U \mathrm{u} + W_P \mathrm{p} + b\right) \tag{11}$$

Where v^{T}, b, W_H, W_U and W_P are parameters which need to be trained.

3.4 Document Representation

Since we have obtained the i-th sentence representation s_i based on pure text, we concatenate s_i with the corresponding user statistic vector $\widetilde{u_r^s}$ and product statistic vector $\widetilde{p_r^s}$ as the full representation for i-th sentence:

$$s'_i = [s_i, \widetilde{u^s_r}, \widetilde{p^s_r}] \tag{12}$$

As the similar computation process on word level, all sentences of a document $\{s'_1, s'_2, ..., s'_n\}$ are fed to hierarchical Bi-LSTM network with attention mechanism on sentence level to obtain document representation $\hat{d}$.

$$\hat{d} = \sum_{i=1}^{n} \beta_i s'_i \tag{13}$$

Where β_i means sentence level attention weight. Sentence level attention shares the same mechanism with word level attention but with different parameters.

3.5 Sentiment Classification

Since $\hat{d}$ is a high level representation of the document semantic representation. Hence, we regard it as features for sentiment classification. We use a non-linear layer to project document semantic representation $\hat{d}$ into the target space of C classes:

$$h_k = \tanh(W_c \hat{d} + b_c) \tag{14}$$

Softmax function is followed to obtain the sentiment distribution:

$$p_c = \frac{\exp(h_c)}{\sum_{k=1}^{C} \exp(h_k)} \tag{15}$$

Where C is the number of sentiment classes; p_c is the predicted probability of sentiment class c. In our model, cross-entropy error between gold sentiment distribution and predicted sentiment distribution is defined as loss function for optimization while training:

$$cost = -\sum_{d \in D} \sum_{c=1}^{C} t_c(d) \log(p_c(d)) \tag{16}$$

Where $t_c(d)$ is the gold probability of sentiment class c with ground truth being 1 and others being 0 and D is the training documents set.

4 Experiment

4.1 Dataset

We evaluate the validity of our model with three popular movie review datasets: IMDB, yelp13 and yelp14, which have been widely used for sentiment analysis research. The sentiment label of IMDB dataset ranges from 1 to10. The sentiment labels of Yelp 2014 and Yelp 2013 datasets range from 1 to 5. The higher label value is, the more positive it represents. Yelp datasets are collected form Yelp site, which is the largest review websites in America including larger number of movie reviews. IMDB dataset is collected from Internet Movie Database, which is the world's most popular and authoritative source for movie including rich reviews.

The statistical information of the three datasets is shown in Table 1.

Dataset	#cla-sses	#users	#pro-ducts	#rev-iews	#docs /user	#docs /product	#sents /doc	#words /sen	#words /doc
IMDB	10	1,310	1,635	84,919	64.82	51.94	16.08	24.54	394.6
Yelp 2014	5	4,818	4,194	231,163	47.97	55.11	11.41	17.26	196.9
Yelp 2013	5	1,631	1,633	78,966	48.42	48.36	10.89	17.38	189.3

Table 1. Statistical information of the three datasets

We use the dataset setting as the same with [1] which split training, validation and test sets in the proportion of 8:1:1 (80% for training, 10% for test, 10% for validation). Validation set is used to find the optimal parameters for model.

4.2 Metrics

We employ standard accuracy rate to measure the overall sentiment classification performance. And we also employ Root-Mean-Square Error (RMSE) to measure the divergences between predicted sentiment label and ground truth label. The higher accuracy and the smaller RMSE is, the better performance is.

4.3 Baselines

In order to fully assess the efficiency of our model, we compare with the following methods, which are widely used as baselines in other sentiment analysis works [5, 9, 10].

- Majority Method [7] .
  Refer to the [7] , the widely used in the method of sentiment classification.
- Supported Vector Machine (SVM) classifiers
  This class of methods employs different features to train a SVM classifier.
- UPNN [3]
  Embeds user and product into high dimensions vectors and utilizing CNN to predict the sentiment label.
- RNTN + RNN [6]
  This model takes phrases of any length as input, using RNN to obtain document representation for sentiment classification
- NSC [4]
  NSC is a hierarchical LSTM model for classify sentiment level of a document.
- NSC+UPA [4]
  NSC+UPA gives excellent performance on document level sentiment classification task.

4.4 Experiment Result

Table 2 gives the experiment results of all the models. Acc.(Accuracy) and RMSE are the evaluation metrics mentioned above. The best performances are in bold.

Models	IMDB		Yelp2013		Yelp2014	
	Acc.	RMSE	Acc.	RMSE	Acc.	RMSE
Majority	0.196	2.495	0.411	1.060	0.392	1.097
RNTN + RNN	0.400	1.764	0.582	0.821	0.574	0.804
SVM-ngram	0.399	1.783	0.569	0.814	0.577	0.804
SVM-TextFeature	0.402	1.793	0.556	0.845	0.572	0.800
SVM-AvgWordvec	0.304	1.985	0.526	0.898	0.530	0.893
SVM-SSWE	0.312	1.973	0.549	0.849	0.557	0.851
UPNN (CNN and no UP)	0.405	1.629	0.577	0.812	0.585	0.808
UPNN (Full)	0.435	1.602	0.596	0.784	0.608	0.764
NSC	0.443	1.465	0.627	0.701	0.637	0.686
NSC+LA	0.487	1.381	0.631	0.706	0.630	0.715
NSC+UPA	0.533	1.281	0.650	0.692	0.667	0.654
NCSI	**0.553**	**1.174**	**0.661**	**0.672**	**0.670**	**0.643**

Table 2. Experiment results of all the models

Models	IMDB		Yelp2013		Yelp2014	
	Acc.	RMSE	Acc.	RMSE	Acc.	RMSE
W/O up_sta	0.512	1.299	0.632	0.713	0.656	0.666
W/O pro_sta	0.541	1.291	0.657	0.668	0.665	0.657
W/O user_sta	0.520	1.230	0.637	0.687	0.670	0.647
Full model	**0.553**	**1.174**	**0.661**	**0.672**	**0.670**	**0.643**

Table 3. Effect of Statistical Information

From Table 2, we can see that our model outperforms all other methods on all datasets in term of both metrics.

It is easy to understand that Majority has a poor performance because it only uses rough statistics and ignores any semantic information.

We can see that the performance of SVM based methods is not good as methods based on neural networks. RNTN+RNN is effective in modelling document representation with semantic composition. It achieves comparable performance with the best result of SVM series. UPNN uses convolution neural network to capture text information and performs slightly better than SVM series and RNTN+RNN. UPNN(full) takes user and product information and improves the performance of UPNN.

NSC model gets excellent performance because it captures the sequence text information via using LSTM architecture, which is good at remembering information for long periods. NSC+LA brings improvement via introducing local attention mechanism. NSC+UPA introduces user and product information to NSC via attention mechanism and significantly boosts the performance.

The results of UPNN series and NSC series have proven that the use of user and product information can improve the performance.

In term of accuracy rate, we can see that our model outperforms previous best model NSC+UPA 2%, 1.1% and 0.3% on three datasets respectively.

In term of RMSE, our model outperforms previous best model NSC+UPA 0.107, 0.02 and 0.011 over three datasets respectively.

The improvement can be interpreted that user and product information is employed as their words' embeddings in other works. But our approach is more focused on using their statistic information. As a result, our model is effective to capture more information including both semantic and statistic, which is crucial for sentiment classification.

In sum, we can make a safe conclusion that our model improves robustness and performance of neural classifier on sentiment analysis task via taking user and product statistic information into account.

4.5 Ablation Study: Effect of Statistic Information

The ablation study is performed to evaluate whether and how each component of our model contributes to our full model.

We ablate two important components and conduct different approaches in this experiment.

W/O user_sta, refers to no user statistic information is employed.

W/O pro_sta, refers to no product statistic information is employed.

W/O up_sta, refers to neither of user and product statistic information is employed.

Table 3 gives the results that explain the effect brought by statistic information proposed in our work.

From table 3, we can see that statistic information employed brings improvement on sentiment classification. Each statistic information can improve the result with different degrees in terms of two metrics. Model without using any statistic information performs worst.

Between two components, user statistic information brings a little bigger improvement. This is probably due to the fact that the sentiment is expressed by the user who directly determines the final sentiment. Product is objective entity while user's review is subjective analysis. So the user information may play more important role. Using both statistic information (full model) achieves the best performance.

5 Conclusion

In this work, we propose a novel neural classifier with statistic information for sentiment analysis on document level. Our model extracts and feeds statistic information carried by user and product to neural networks. It can utilize user preference and product characteristics and yield better semantic representation. Experiment results on three popular datasets have shown that our model achieves state-of-the-art. This paper gives a new way for sentiment analysis. Our future work will explore more statistic information which can be used to boost the performance on sentiment analysis. And another interesting future work is to extend this model to aspect level sentiment analysis.

References

1. Tagging, Domain-Sensitive Temporal. "Synthesis Lectures on Human Language Technologies."
2. Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014).
3. Tang, Duyu, Bing Qin, and Ting Liu. "Learning semantic representations of users and products for document level sentiment classification." Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Vol. 1. 2015.
4. Chen, Huimin, et al. "Neural sentiment classification with user and product attention." Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 2016.
5. Li, Changliang, et al. "Parallel recursive deep model for sentiment analysis." Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, Cham, 2015.
6. Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." Proceedings of the 2013 conference on empirical methods in natural language processing. 2013.
7. Tang, Duyu, Bing Qin, and Ting Liu. "Aspect level sentiment classification with deep memory network." arXiv preprint arXiv:1605.08900 (2016).
8. Maas, Andrew L., Awni Y. Hannun, and Andrew Y. Ng. "Rectifier nonlinearities improve neural network acoustic models." Proc. icml. Vol. 30. No. 1. 2013.
9. Tang, Duyu, et al. "Target-dependent sentiment classification with long short term memory." CoRR, abs/1512.01100 (2015).
10. Wang, Yequan, Minlie Huang, and Li Zhao. "Attention-based lstm for aspect-level sentiment classification." Proceedings of the 2016 conference on empirical methods in natural language processing. 2016.