Capturing User and Product Information for Document Level Sentiment Analysis with Deep Memory Network

Zi-Yi Dou
National Key Laboratory for Novel Software Technology, Nanjing University,
Nanjing, 210023, China
141242042@smail.nju.edu.cn

Abstract

Document-level sentiment classification is a fundamental problem which aims to predict a user's overall sentiment about a product in a document. Several methods have been proposed to tackle the problem whereas most of them fail to consider the influence of users who express the sentiment and products which are evaluated. To address the issue, we propose a deep memory network for document-level sentiment classification which could capture the user and product information at the same time. To prove the effectiveness of our algorithm, we conduct experiments on IMDB and Yelp datasets and the results indicate that our model can achieve better performance than several existing methods.

1 Introduction

Sentiment analysis, sometimes known as opinion mining, is the field of study that analyzes people's opinions, sentiments, evaluations, attitudes and emotions from written language. It is one of the most active and critical research areas in natural language processing (Liu, 2012). On the one hand, from the industry point of view, knowing the feelings among consumers based on their comments is beneficial and may support strategic market decisions. On the other hand, potential customers are often interested in other people's opinion in order to find out the choices that best fits their preferences (Moraes et al., 2013).

Previous studies tackled the sentiment analysis problem at various levels of granularity, from document level to sentence level due to different objectives of applications (Zhang et al., 2009). In this work, we mainly focus on document-level sentiment classification Basically, the task is to predict user's overall sentiment or polarity in a document about a product (Pang and Lee, 2008).

Most existing methods mainly utilize local text information whereas ignoring the influences of users and products (Tang et al., 2015). As is often the case, there are certain consistencies for both users and products. To illustrate, lenient users may always give higher ratings than fastidious ones even if they post the same review. Also, it is not surprising that some products may always receive low ratings because of their poor quality and vice versa. Therefore, it is necessary to leverage individual preferences of users and overall qualities of products in order to achieve better performance.

Tang *et al.* (2015) proposed a novel method dubbed User Product Neural Network (UPNN) which capture user- and product-level information for sentiment classification. Their approach has shown great promise but one major drawback of their work is that for users and products with limited information, it is hard to train the representation vector and matrix for them.

Inspired by the recent success of computational models with attention mechanism and explicit memory (Graves et al., 2014; Sukhbaatar et al., 2015), we addressed the aforementioned issue by proposing a method based on deep memory network and Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). The model can be divided into two separate parts. In the first fart, we utilize LSTM to represent each document. Afterwards, we apply deep memory network consists of multiple computational layers to predict the ratings for each document and each layer is a content-based attention model.

To prove the effectiveness of our algorithm, we have conducted experiments on three datasets derived from IMDB and Yelp Dataset Challenge and compare to several other algorithms. Experimental results show that our algorithm can outperform

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 521–526
Copenhagen, Denmark, September 7–11, 2017.

baseline methods for sentiment classification of documents by leveraging users and products for document-level sentiment classification.

2 Related Work

2.1 Memory Network

In 2014, Weston *et al.* (2014) introduced a new class of learning models called memory networks. Memory networks reason with inference components combined with a long-term memory component. The long-term memory can be read and written to and then it can be used for prediction. Generally, a memory network consists of an array of objects called memory m and four components I, G, O and R, where I converts input to internal feature representation, G updates old memories, O generates an output representation and R outputs a response.

Based on their work, Sukhbaatar *et al.* (2015) proposed a neural network with a recurrent attention model over a possibly large external memory. Unlike previous model, their model is trained end-to-end and hence requires significantly less supervision during training. They have shown that their model yields improved results in language model and question answering.

Inspired by the success of memory network, Tang *et al.* (2016) introduce a deep memory network for aspect-level sentiment classification. The architecture of their model is similar to the previous model and experimental results demonstrate that their approach performs comparable to other state-of-the-art systems. Also, Li *et al.* (2017) decompose the task of attitude identification into two separate subtasks: target detection and polarity classification; and then solve the problem by applying deep memory network so that signals produced in target detection provide clues for polarity classification and the predicted polarity provides feedback to the identification of targets.

2.2 Sentiment Classification

Most existing work tackle the problem of sentiment classification by manually design effective features. such as text topic (Ganu et al., 2009) and bag-of-opinion (Qu et al., 2010) . Some work take user information into consideration. For example, in 2013, Gao *et al.* (2013) design user-specific features to capture user leniency. Also, Li *et al.* (2014) incorporate textual topic and user-word factors with supervised topic modeling.

Tang *et al.* (2015) points out that it is critical to leverage users and products for document-level sentiment classification. They assume there are four types of consistencies for sentiment classification and validate the influences of users and products in terms of sentiment and text on massive IMDB and Yelp reviews. Their model represent each user and product as both vector and matrix in order to capture the consistencies and then apply convolutional neural network to solve the task.

To the best of our knowledge, no one has ever applied deep memory network to capture the user and product information and solve the tasks in sentiment classification at document-level.

3 Proposed Methods

In this section, we present the details of User Product Deep Memory Network (UPDMN) for sentiment classification at document level.

3.1 Basic Symbol and Definition

First we suppose U, P, D is the set of users, products and documents respectively. If user $u \in U$ writes a document $d \in D$ about a product $p \in P$ and give the rating, we denote $U(d) = \{ud|ud \text{ is written by } u, ud \neq d\}$ and $P(d) = \{pd|pd \text{ is written about } p, pd \neq d\}$. Then, our task can be formalized as follows: suppose u write a document d about a product p , we should output the predicted score y for the document d based on the input $< d, U(d), P(d) >$. The detail of these symbols would be illustrated in the following part.

3.2 General Framework of UPDMN

Figure 1 illustrates the general framework of our approach. Basically, inspired by the use of memory network in question answering and aspect-level sentiment analysis (Sukhbaatar et al., 2015; Tang et al., 2016), our model consists of multiple computational layers (hops), each of which contains an attention layer and a linear layer.

For every document in $U(d)$ and $P(d)$, we embed it into a continuous vector d_i and store it in the memory. The model writes all document to the memory up to a fixed buffer size. Suppose we are given $\{d_i\} = \{d_1, ..., d_n\}$ to be stored in memory, for each layer we can convert them into memory vectors $\{m_i\}$ using an embedding matrix. The document d should also be embedded into q. Then, we compute the match $\{p_i\}$ between q and each memory m_i. Afterwards, we embed $\{d_i\}$ into

Dataset	#users	#products	#reviews	#docs/user	#docs/product	#sents/doc	#words/doc
IMDB	1,310	1,635	84,919	64.82	51.94	16.08	394.6
Yelp 2014	4,818	4,194	231,163	47.97	55.11	11.41	196.9
Yelp 2013	1,631	1,633	78, 966	48.42	48.36	10.89	189.3

Table 1: Statistical information of datasets.

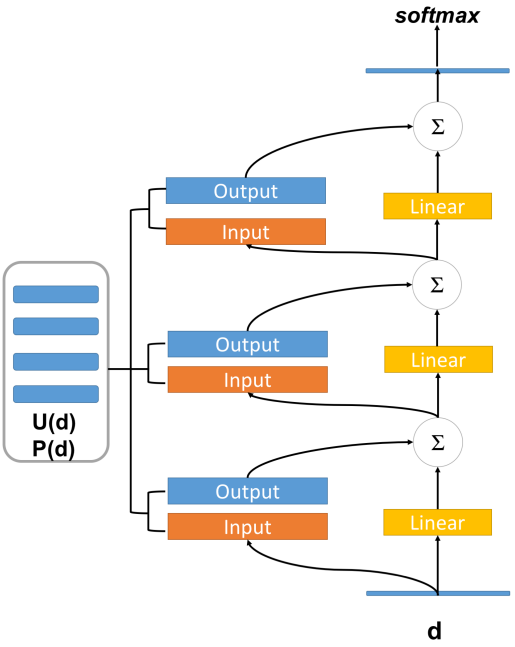

Figure 1: General Framework

output vector $\{c_i\}$ using another embedding matrix and generate the output of attention layer. The output is further summed with the linear transformation of q and considered as the input of next hop. The output vector at last hop is fed into a *softmax* layer and then generates the final prediction y for document-level sentiment classification.

3.3 Embedding Documents

Although there are several state-of-the-art techniques to embed word into vectors (Mikolov et al., 2013a), for document-level sentiment classification, the document we need to classify is usually too long to be represented as a vector. People have tried different ways to solve the task. For example, Kalchbrenner *et al.* (2014) apply convolutional neural network for modeling sentences and Li *et al.* (2015) introduce an LSTM model that hierarchically builds an embedding for a paragraph from embeddings for sentences and words.Some of these work can be incorporated into our methods. However, here we only use the LSTM model to embed each document, *i.e.* every word in the document is fed into LSTM and the final representation is obtained by averaging the hidden state of each word, and the experimental results shows that this simple embedding method can actually obtain satisfactory results.

3.4 Attention Model

After obtaining the embedding vector q for document d the memory vectors $\{m_i\}$ for each memory, we calculate the match between q and m_i using the following equation:

$$p_i = softmax(W_{att}[m_i; q] + b_{att}) \quad (1)$$

where $softmax(z_i) = e^{z_i} / \sum_j e^{z_j}$.

Afterwards, we compute the corresponding output o for each hop by summing over the c_i, weighted by the probability vector from the input:

$$o = \sum_i p_i c_i \quad (2)$$

3.5 Final Prediction and Training Strategy

At last hop, the output vector is fed into a *softmax* layer and thus generates a probability distribution $\{y_i\}$ over ratings. The score with the highest probability would be considered as our final prediction py. During training, we try to minimize the cross entropy error of sentiment classification in a supervised manner. The specific equation is shown as follows:

$$Loss = -\sum_{d \in D} \sum_{y_i \in Y} \mathbb{I}(y = y_i|d) log(P(y = y_i|d)) \quad (3)$$

where Y is the collection of sentiment categories, $\mathbb{I}(y = y_i|d)$ is 1 or 0, indicating whether the correct category for d is y_i, and $P(y = y_i|d)$ represents the probability of classifying document d as category y_i.

4 Experiment

In this section, we will first discuss the experimental setting and then display the results.

	IMDB			Yelp 2014			Yelp 2013		
	Acc	MAE	RMSE	Acc	MAE	RMSE	Acc	MAE	RMSE
Majority	0.196	1.838	2.495	0.392	0.779	1.097	0.411	0.744	1.060
Trigram	0.399	1.147	1.783	0.577	0.487	0.804	0.569	0.513	0.814
TextFeature	0.402	1.134	1.793	0.572	0.490	0.800	0.556	0.520	0.845
AvgWordvec + SVM	0.304	1.361	1.985	0.530	0.562	0.893	0.526	0.568	0.898
SSWE + SVM	0.312	1.347	1.973	0.557	0.523	0.851	0.549	0.529	0.849
Paragraph Vector	0.341	1.211	1.814	0.564	0.496	0.802	0.554	0.515	0.832
RNTN + Recurrent	0.400	1.133	1.764	0.582	0.478	0.821	0.574	0.489	0.804
Trigram + UPF	0.404	1.132	1.764	0.576	0.471	0.789	0.570	0.491	0.803
TextFeature +UPF	0.402	1.129	1.774	0.579	0.476	0.791	0.561	0.509	0.822
JMARS	N/A	1.285	1.773	N/A	0.710	0.999	N/A	0.699	0.985
UPNN	0.435	0.979	1.602	0.608	0.447	0.764	0.596	0.464	0.784
UPDMN(1)	0.428	0.936	1.443	0.588	0.457	0.757	0.596	0.454	0.747
UPDMN(2)	0.446	0.944	1.495	0.592	0.451	0.748	0.602	0.437	0.732
UPDMN(3)	0.459	0.883	1.397	0.599	0.444	0.742	0.627	0.386	0.681
UPDMN(4)	**0.465**	**0.853**	**1.351**	0.609	0.432	0.731	**0.639**	**0.369**	**0.662**
UPDMN(5)	0.456	0.928	1.471	**0.613**	**0.425**	**0.720**	0.611	0.405	0.704

Table 2: Experimental results.

4.1 Experimental Settings

We use the same datasets as Tang *et al.* (2015), which are derived from IMDB (Diao et al., 2014) and Yelp Dataset Challenge in 2013 and 2014 [1]. Statistical information of the datasets are given in Table 1.

In order to measure the performance of our model, here we use three metrics. Specifically, we use $accuracy$ to measure the overall sentiment classification performance, MAE and $RMSE$ to measure the divergences between prediction py and ground truth gy. The formulas for these three metrics are listed as follows:

$$accuracy = \frac{T}{N} \tag{4}$$

$$MAE = \frac{\sum_i |py_i - gy_i|}{N} \tag{5}$$

$$accuracy = \sqrt{\frac{\sum_i (py_i - gy_i)^2}{N}} \tag{6}$$

4.2 Baseline Models

We compare UPDMN with the following models:

(1) **Majority** : it assigns each review in the test dataset with the majority sentiment category in training set.

(2) **Trigram** : it first takes unigrams, bigrams and trigrams as features and then trains a classifier with SVM (Fan et al., 2008).

(3) **TextFeature** : it takes hard-crafted text features such as word/character n-grams, negation features and then trains a classifier with SVM.

(4) **UPF**: it extracts user-leniency features and corresponding product features from training data and then concatenates them with features in model (2) and (3) (Gao et al., 2013).

(5) **AvgWordvec+SVM** : it learns word embeddings from training and development sets with $word2vec$, averages word embeddings and then trains an SVM classifier (Mikolov et al., 2013b).

(6) **SSWE+SVM** : it learns sentiment-specific word embeddings (SSWE), uses max/min/average pooling to generate document representation and then trains an SVM classifier (Tang et al., 2014).

(7) **RNTN+RNN** : it represents each sentence with RNTN, composes document with recurrent neural network , and then averages hidden vectors of recurrent neural network as the features (Socher et al., 2013).

(8) **Paragraph Vector**: it implements the PVDM for document-level sentiment classification (Le and Mikolov, 2014).

(9) **JMARS**: it is the recommendation algorithm which leverages user and aspects of a review with collaborative filtering and topic modeling (Diao et al., 2014).

(10) **UPNN** : as has been stated above, it also leverages user and product information for sentiment classification at document level (Tang et al., 2015).

4.3 Experimental Results and Discussion

The experimental results are given in Table 2. The results of baseline models are reported in (Tang et al., 2015). Our model is abbreviated to UPDMN(k), where k is the number of hops. With the increase of the number of hops, the performance of UPDMN will get better intially, which indicates that multiple hops can indeed capture

[1] http://www.yelp.com/dataset_challenge

more information to improve the performance. However, if there are too many hops, the performance would be not as well as before, which may be caused by over-fitting.

Compared with other models, we can see that with proper setting, our model achieve superior results. All these results prove the effectiveness of UPDMN and the necessity to utilizing user and product information at document level.

It should be noticed that there are still several improvements can be made, such as better representation of documents or more sophisticated attention mechanism. We believe that our model has great potential and can be improved in many ways.

Acknowledgments

We thank anonymous reviewers and Hao Zhou for helpful advice.

References

Qiming Diao, Minghui Qiu, Chao Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, pages 193–202.

Rong En Fan, Kai Wei Chang, Cho Jui Hsieh, Xiang Rui Wang, and Chih Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9(9):1871–1874.

Gayatree Ganu, Noemie Elhadad, and Amlie Marian. 2009. Beyond the stars: Improving rating predictions using review text content. In *International Workshop on the Web and Databases, WEBDB 2009, Providence, Rhode Island, Usa, June*.

Wenliang Gao, Naoki Yoshinaga, Nobuhiro Kaji, and Masaru Kitsuregawa. 2013. Modeling user leniency and product popularity for sentiment classification. In *IJCNLP*, pages 1107–1111.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *Computer Science*.

Seppu Hochreiter and Jrgen Schmidhuber. 1997. *Long short-term memory*. Springer Berlin Heidelberg.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Eprint Arxiv*, 1.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *Computer Science*, 4:1188–1196.

Fangtao Li, Sheng Wang, Shenghua Liu, and Ming Zhang. 2014. Suit: a supervised user-item based topic model for sentiment analysis. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1636–1642.

Jiwei Li, Minh Thang Luong, and Jurafsky Dan. 2015. A hierarchical neural autoencoder for paragraphs and documents. *Computer Science*.

Bing Liu. 2012. Sentiment analysis and opinion mining. In *Synthesis Lectures on Human Language Technologies*, page 167.

Qiaozhu Mei, Qiaozhu Mei, and Qiaozhu Mei. 2017. Deep memory networks for attitude identification. pages 671–680.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119.

Rodrigo Moraes, Jo Valiati, and Wilson P O Neto. 2013. Document-level sentiment classification: An empirical comparison between svm and ann. *Expert Systems with Applications*, 40(2):621–633.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(12):1–135.

Lizhen Qu, Georgiana Ifrim, and Gerhard Weikum. 2010. The bag-of-opinions method for review rating prediction from sparse text patterns. In *COLING 2010, International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China*, pages 913–921.

R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. *Computer Science*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, pages 1014–1023.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Conference on Empirical Methods in Natural Language Processing*, pages 214–224.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Meeting of the Association for Computational Linguistics*, pages 1555–1565.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *Eprint Arxiv*.

Changli Zhang, Daniel Zeng, Jiexun Li, Fei Yue Wang, and Wanli Zuo. 2009. Sentiment analysis of chinese documents: From sentence to document level. *Journal of the Association for Information Science and Technology*, 60(12):2474–2487.