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 opinions 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 part, 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
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
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 show 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:

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

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:

\[
o = \sum_i p_i c_i
\]

### 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:

\[
Loss = - \sum_{d \in D} \sum_{y \in Y} I(y = y_i | d) \log(P(y = y_i | d))
\]

where \( Y \) is the collection of sentiment categories, \( I(y = y_i | d) \) 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.
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:

\[
\text{accuracy} = \frac{T}{N} \quad (4)
\]

\[
\text{MAE} = \frac{\sum |py_i - gy_i|}{N} \quad (5)
\]

\[
\text{RMSE} = \sqrt{\frac{\sum (py_i - gy_i)^2}{N}} \quad (6)
\]

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 initially, which indicates that multiple hops can indeed capture

Table 2: Experimental results.

<table>
<thead>
<tr>
<th></th>
<th>IMDB</th>
<th>Yelp 2014</th>
<th>Yelp 2013</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc</td>
<td>MAE</td>
<td>RMSE</td>
</tr>
<tr>
<td>Majority</td>
<td>0.196</td>
<td>1.838</td>
<td>2.495</td>
</tr>
<tr>
<td>Trigram</td>
<td>0.399</td>
<td>1.147</td>
<td>1.783</td>
</tr>
<tr>
<td>TextFeature</td>
<td>0.402</td>
<td>1.134</td>
<td>1.793</td>
</tr>
<tr>
<td>AvgWordvec+SVM</td>
<td>0.304</td>
<td>1.361</td>
<td>1.985</td>
</tr>
<tr>
<td>SSWE+SVM</td>
<td>0.312</td>
<td>1.347</td>
<td>1.973</td>
</tr>
<tr>
<td>Paragraph Vector</td>
<td>0.341</td>
<td>1.211</td>
<td>1.814</td>
</tr>
<tr>
<td>RNTN+Recurrent</td>
<td>0.400</td>
<td>1.133</td>
<td>1.764</td>
</tr>
<tr>
<td>Trigram + UPF</td>
<td>0.404</td>
<td>1.132</td>
<td>1.764</td>
</tr>
<tr>
<td>TextFeature +UPF</td>
<td>0.402</td>
<td>1.129</td>
<td>1.774</td>
</tr>
<tr>
<td>JMARS</td>
<td>N/A</td>
<td>1.285</td>
<td>1.773</td>
</tr>
<tr>
<td>UPNN</td>
<td>0.435</td>
<td>0.979</td>
<td>1.602</td>
</tr>
<tr>
<td>UPDMN(1)</td>
<td>0.428</td>
<td>0.936</td>
<td>1.443</td>
</tr>
<tr>
<td>UPDMN(2)</td>
<td>0.446</td>
<td>0.944</td>
<td>1.495</td>
</tr>
<tr>
<td>UPDMN(3)</td>
<td>0.459</td>
<td>0.883</td>
<td>1.397</td>
</tr>
<tr>
<td>UPDMN(4)</td>
<td>0.465</td>
<td>0.853</td>
<td>1.351</td>
</tr>
<tr>
<td>UPDMN(5)</td>
<td>0.456</td>
<td>0.928</td>
<td>1.471</td>
</tr>
</tbody>
</table>

1http://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


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.
