Learning from User Social Relation for Document Sentiment Classification

Kangzhi Zhao(✉), Yong Zhang, Yan Zhang, Chunxiao Xing, and Chao Li

RIIT, Beijing National Research Center for Information Science and Technology, Department of Computer Science and Technology, Institute of Internet Industry, Tsinghua University, Beijing 100084, China
{zkz15,zhang-yan14}@mails.tsinghua.edu.cn,
{zhangyong05,xingcx,li-chao}@tsinghua.edu.cn

Abstract. Sentiment analysis is a fundamental problem in the field of natural language processing. Existing methods incorporate both semantics of texts and user-level information into deep neural networks to perform sentiment classification of social media documents. However, they ignored the relations between users which can serve as a crucial evidence for classification. In this paper, we propose SRPNN, a deep neural network based model to take user social relations into consideration for sentiment classification. Our model is based on the observation that social relations between users with similar sentiment trends provide important signals for deciding the polarity of words and sentences in a document. To make use of such information, we develop a user trust network based random walk algorithm to capture the sequence of users that have similar sentiment orientation. We then propose a deep neural network model to jointly learn the text representation and user social interaction. Experimental results on two popular real-world datasets show that our model significantly outperforms state-of-the-art methods.

1 Introduction

With the popular social media such as microblog services and review sites, users can conveniently share their personal feelings and opinions on the Internet, and embed their characteristics and preferences into the subjective text [32]. Given a collection of documents, the task of sentiment classification is to infer the sentiment polarity or intensity of each document. With the rapid growth of social media data, sentiment classification has drawn much attention from research communities in recent years [23,25,33], which arises in many real world applications such as opinion mining, personalized recommendation and market analysis.

Early studies in this area mainly adopt feature-based method and construct classifiers to solve this problem. Pang and Lee [18] first adopted supervised learning method to build classifiers. Many studies [10,12,21] tried to integrate various types of features to enhance the effectiveness. Despite the plausible success of some shallow learning methods, feature engineering is labor-intensive.

G. Li et al. (Eds.): DASFAA 2019, LNCS 11447, pp. 86–103, 2019.
https://doi.org/10.1007/978-3-030-18579-4_6

Many models need domain-specific features, which make it difficult apply them to other datasets or applications.

Recently there have been some neural network based studies to extract features automatically so as to avoid the complicated feature engineering. Some previous studies focus on designing effective models for classification [2,22], while others aim at learning better representations of the text [9,28]. Recent studies [27,33] further integrate features other than text representation, such as user personality and product information into the model to enhance the effectiveness. However, such studies also suffer from the data sparsity problem. For example, in the product review rating, one product or topic has only a few reviews from a user, which makes it difficult to develop an accurate predication of the sentiment.

To address this problem, we argue that the social relations between users can be adopted to augment the data so as to provide important signals for sentiment analysis. Our idea is based on two observations. Firstly, users have specific preferences on providing sentiment ratings. And users with similar sentiment orientation tend to have similar comments on one product or event. For example, if a user posts a tweet saying "Trump is the one who changes America", it is difficult to judge his sentiment trend towards Trump. However, if we know that he has many followers who are against Trump, we can infer that it is very likely that he is against Trump, too. Secondly, a user with high authority ("opinion leader") provides strong signals on the sentiment. For example, a user complains that his cellphone is easily overheating. He praises this brand of cellphones as "good at its warmth in winter". It is difficult to identify this ironic negative comments from just the texts. However, if the user follows a tech leader who also blames about the overheating of his cellphone in studies, it is easier to obtain the user's sentiment orientation to this cellphone by considering his interactions with the tech leader. Therefore, by constructing **user document** with social relations from above two aspects, the problem of data sparsity in original documents can be extensively alleviated.

In this paper, we propose **S**ocial **R**elation **P**owered **N**eural **N**etwork (SRPNN) model to utilize the user social relations for document-level sentiment classification. We first model user social relations as a user trust network and then propose a random walk algorithm to generate user documents from the network based on both user authority and sentiment similarity. We then propose a deep neural network model which exploits both the semantic representation of texts and user document to predict the sentiment orientation. To the best of our knowledge, this is the first work to jointly learn text representation and user social relations for sentiment classification.

We evaluate SRPNN on two popular datasets Twitter and Yelp and compare it with several state-of-the-art methods. Experimental results show that SRPNN outperforms various baseline methods including both feature-based approaches and deep learning based method. It also demonstrated the effectiveness of incorporating user social relations. The contributions of this paper are summarized as following:

- We propose a novel model SRPNN by leveraging the user social relations for document-level sentiment classification. To the best of our knowledge, this is the first work that combines the text representation and features of user social relation as the input of deep neural networks.
- We design a random walk based algorithm to generate high quality user document considering both user authority and sentiment similarity.
- We conduct extensive sets of experiments on two popular datasets. The experimental results demonstrate the effectiveness of our model.

2 Related Work

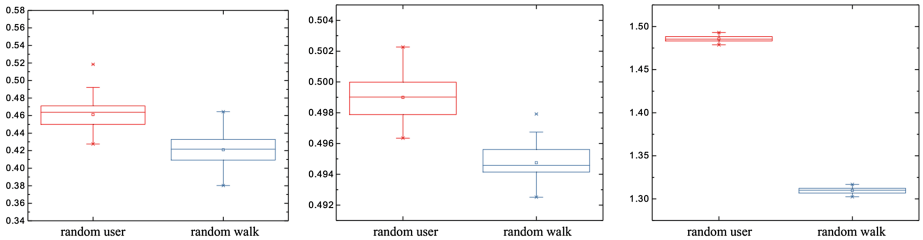
Trust Learning in User Network. Random Walk is an algorithm that generates a sequence of visited nodes by iteratively selecting a random neighbor of current node. It has been widely used in the applications like collaborate filtering [3] and personalized recommendation [14]. A large number of studies also adopt the idea of random walk [29,35]. DeepWalk [19] adopts the neural language model to learn the embedding of network and terminates the random walk sequence by setting a maximum step size. TidalTrust [4] utilizes a BFS algorithm to search the trust score between users in a network. TrustWalker [6] proposed a random walk based framework for recommendation problems.

Sentiment Classification. There is a long stream of studies for sentiment classification on documents. Pang and Lee [18] proposed a supervised learning framework for sentiment classification. Many studies design rich features to enhance the effectiveness, such as bag of opinion [21], product information [10] and sentiment lexicon [7]. Some studies [12] focused on integrating emotional signals into machine learning framework for sentiment analysis. Hu et al. [5] utilized matrix manipulations to address the noises in microblog texts and construct sentiment relations. Zhu et al. [38] focused on improving the efficiency of sentiment analysis in large scale of social networks.

Many studies adopted data-driven approaches to avoid handcrafted features [30,36]. Mikolov et al. [15] utilized the context information to train the word and phrase embedding. Le and Mikolov [9] introduced paragraph embedding. Socher et al. [22] and Dong et al. [2] proposed recursive deep neural networks for sentiment classification. Mishra et al. [16] adopted convolutional neural network, while Tang et al. [24] and Qian et al. [20] adopted recurrent neural network for sentiment analysis. Recently attention mechanism [17] is also widely used in multiple NLP tasks especially in sentiment classification [11,13,34].

Personalized Sentiment Classification. Personalized Sentiment Classification has become a popular topic recently. Tang et al. [28] incorporated sentiment information when learning the word embedding. Tang et al. [27] obtained richer feature for neural network by modeling the personality of users. They further integrated the product information and the aspect level information to help improve the effect of classification [25,26]. Chen et al. [1] adopted selectivity attention to model the relation between users and products. Song et al. [23] adopted user's

following relation matrix to extend Latent Factor Model in microblog sentiment classification. Wu and Huang [33] constructed a personalized classifier to integrate user's social network to sentiment classification. Zhao et al. [37] introduced a network embedding learning framework on heterogeneous microblog network. Wang et al. [31] incorporated user's cross-lingual sentiment consistency with a multi-task learning framework to enrich the user post representation.



(a) Twitter_1000 (b) Twitter (c) Yelp

Fig. 1. User text-sentiment consistency for random walk

3 Constructing User Relation Sequence

In this section, we introduce the model to construct user relation sequences. We first justify the rationality to generate a series of user sequences using random walk and propose a user trust network to model the social similarity between users. We then take advantage of the network and devise a trust score as the metric to generate user relation sequence. Finally we propose a random walk based algorithm to effectively obtain the user relation sequences.

3.1 User-Sentiment Consistency Verification

We have the observation that one user tends to produce documents with similar sentiment orientation. For example, a harsh user tends to evaluate the weaknesses of products, while an amiable user may focus on the advantages. Following this route, we find that users with similar sentiment orientation always tend to have similar comments on one product or event. We further argue that the sequence of user with user-sentiment consistency can be obtained using random walk algorithm. This can serve as the foundation of our model. Next we will validate this assumption.

To test the consistency assumptions of the random walk algorithm, for each user in Twitter and Yelp datasets we test for $n = 50$ iterations. Given a user u_i, we randomly pick out one review rating r_j. If u_i has adjacent users, we pick another review rating r_j^{rw} from a user u_i^{rw} in the sequence of u_i in the similar way of executing random walk algorithm. We then randomly pick another

user u_j^{random} and his review rating r_j^{random}. By iteratively calculating the absolute difference (r_j, r_j^{rw}) and (r_j, r_j^{random}), we can see the statistical discrepancy between random walk user and random user.

The test results of all three datasets are shown in Fig. 1. We can see that random walk users hold a lower difference of review rating than random users. Such results confirms the sentiment consistency of the user sequence generated by random walk algorithm.

3.2 User Trust Scoring

Although random walk algorithm can get a better result than selecting random user, we hope to further improve the sentiment consistency. Following the above assumption, we argue that *users with similar sentiment orientation tend to have similar comments on one product or event*. We call such users *trust users* and introduce a user trust network to model their relations. Figure 2 shows an example of the user trust network. The nodes are individual users, with the relationship "user u_i follows user u_j" resulting in an edge directed from node u_i to node u_j. The out-degree of a node denotes the number of people a user follows. The weight $\overline{r}_{u_i}$ on node u_i is its average rating. We utilize the degree of trust $\mathcal{T}(u_i, u_j)$ on the adjacent users in the user trust network to denote the weight of an edge. To describe the degree of trust, we propose a metric named *user trust score*: for users u_i and u_j, the user trust score between them is denoted as $\mathcal{T}(u_i, u_j)$. Users with higher scores will be treated as trust users. The user trust score incorporates the information from two aspects: *user authority* and *sentiment similarity*. Next we will discuss the details about them.

User authority describes how a user is given attention to in the social network. A user with high authority can be regarded as the "opinion leader" in a specific field. If a user pays more attention to opinion leaders, it definitely means that he is interested in a particular topic and shares similar opinions with that user. So opinion leaders should be assigned higher degrees of trust. With the help of user authority, it is easy to find users with common interests.

To quantify user authority, for each user $u_i \in U$, we assign a user authority score denoted as $\mathcal{A}(u_i)$. According to above discussion, an opinion leader with more followers has higher authority, at the same time, followers will also contribute to the authority of opinion leader. Here in the user trust network, the in-degree of a node is the number of followers of a user; while the out-degree of node can be regarded as the influence a user has on other users. Then we can adopt the principle of *PageRank* to calculate the user authority score as shown in Eq. 1.

$$\mathcal{A}(u_i) = \frac{1-\alpha}{|U|} + \alpha\Big(\sum_{u \in N(u_i)} \frac{\mathcal{A}(u)}{L(u_i)}\Big) \tag{1}$$

where $N(u_i)$ denotes the set of nodes that have an edge pointed to u_i, $L(u_i)$ denotes the out-degree of node U_i and α is the damping factor ($\alpha \in (0, 1)$).

The sentiment similarity describes the similarity of sentiment orientation between users. For two users u_i and u_j, we use $\mathcal{S}(u_i.u_j)$ to denote the sentiment similarity score between them. This score is evaluated by the average rating difference between users. And a lower difference means more similar rating orientation. With the help of sentiment similarity scores, we can find the candidate users who hold similar sentiment orientation. Equation 2 shows the sentiment similarity of users.

$$\mathcal{S}(u_i.u_j) = \frac{1}{\left\|\overline{r}_{u_i} - \overline{r}_{u_j}\right\| + 1} \tag{2}$$

where $\overline{r}_{u_i}$ and $\overline{r}_{u_j}$ are the average rating of user u_i and u_j, respectively. From this equation, we can see that the similarity between users is higher when their rating difference is lower.

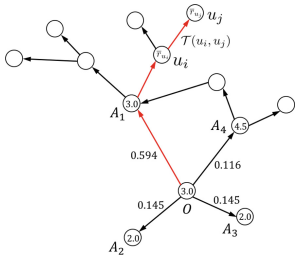


Fig. 2. Toy example of user trust network

Finally we should take both user authority and sentiment similarity into consideration when deciding the user trust score. Intuitively if a user follows a high-authority user, they should be in the same user sequence. But if the sentiment similarity between them is high, putting them together might lead to some deviations in the result. Therefore, given two users u_i and u_j, we propose a hybrid scoring function by combining user authority and sentiment similarity scores of them. The way to calculate $\mathcal{T}(u_i, u_j)$ is shown in Eq. 3.

$$\mathcal{T}(u_i, u_j) = \frac{\mathcal{A}(u_j) \cdot \mathcal{S}(u_i, u_j)}{\sum_{u \in N(u_i)} \mathcal{A}(u) \cdot \mathcal{S}(u_i, u)} \tag{3}$$

We take the user trust network shown in Fig. 2 as an example to illustrate the user trust scoring of node O. Suppose $\overline{r}_O = \overline{r}_{A_1} = 3.0$, $\overline{r}_{A_2} = \overline{r}_{A_3} = 2.0$, $\overline{r}_{A_4} = 4.5$. As shown in Eq. 2, the sentiment similarity between node O and its adjacent node A_i can be calculated as $\mathcal{S}(O, A_1) = 1$, $\mathcal{S}(O, A_2) = \mathcal{S}(O, A_3) = 0.5$, $\mathcal{S}(O, A_4) = 0.4$. With the input of the graph structure (adjacency list), we get the authority score $\mathcal{A}(u)$ of every user, namely, $\mathcal{A}(A_1) = 0.115$, $\mathcal{A}(A_2) = \mathcal{A}(A_3) = \mathcal{A}(A_4) = 0.056$. According to Eq. 3, we normalize $\mathcal{A}(A_i) \cdot \mathcal{S}(O, A_i)$ to get the trust scores $\mathcal{T}(O, A_i)$ of adjacent users as $\mathcal{T}(O, A_1) = 0.594$, $\mathcal{T}(O, A_2) = \mathcal{T}(O, A_3) = 0.145$, $\mathcal{T}(O, A_4) = 0.116$, which are used as the weight of the graph edge to generate user relation sequence from O.

3.3 User-Trust Random Walk Algorithm

Based on above user network, we can construct the user sequences by applying a random walk algorithm on the network structure. In order to include richer information in the user document, we want to include as many users in the sequence as possible. However, if a sequence of users is too long, the trust score will become pretty low, which means a rather low sentiment consistency. To make a trade-off between above factors, we set a stop probability $\phi_{u_i,u_j,k}$ for the random walk algorithm as is shown in Eq. 4. It indicates the probability that u_i stops at u_j after k steps. We also record the maximum number of steps to make sure our algorithm could terminate. Each time when the stop condition is satisfied, we obtain a user relation sequence.

$$\phi_{u_i,u_j,k} = \frac{1}{1+e^{-\frac{k}{2}}} \cdot \frac{\left\|\overline{r}_{u_i} - \overline{r}_{u_j}\right\|}{C} \tag{4}$$

where k is the step number, $\overline{r}_{u_i}$ is the average rating of u_i and C is the number of classification categories.

Input: User set U, user trust network $\mathcal{N}$, maximum step length n
Output: Random walk sequence for all users

```
for u ∈ U do
    k = 0, φ = 0, u_m = u;
    Add u to its own user sequence;
    while k < n and rand(0,1) ≥ φ do
        if u_m has adjacent users then
            Calculate the trust score of u_m for N(u_m);
            Sample the adjacent user in N(u_m) to u_tmp;
            Add u_tmp to the user sequence of u;
            k = k + 1, u_m = u_tmp;
            Update stop probability φ;
        end
    end
end
```

Algorithm 1. Random walk on user trust network

Algorithm 1 shows the process of random walk. For all users in the network, we first initialize their sequences by involving themselves. Then if a user u_m has adjacent users, we will calculate the trust scores between u_m and all the adjacent users using Eq. 4. Next we perform a weighted sampling on the adjacent users, add the samples to u_m's sequence and update the stop probability. We perform above computation iteratively until reaching the maximum step or meeting the stop probability.

4 Deep Learning Based Personalized Sentiment Classification

4.1 Overall Architecture

In this section, we proposed the deep neural network model SRPNN for personalized sentiment classification as is shown in Fig. 3. This model consists of two subnetworks: the left branch captures the features from semantics of review texts. The right branch is a CNN that models the user relations generated by the user trust network introduced in Sect. 3. Both branches consist of three layers: Word Representation Layer, Sentence Representation Layer and Document Representation Layer.

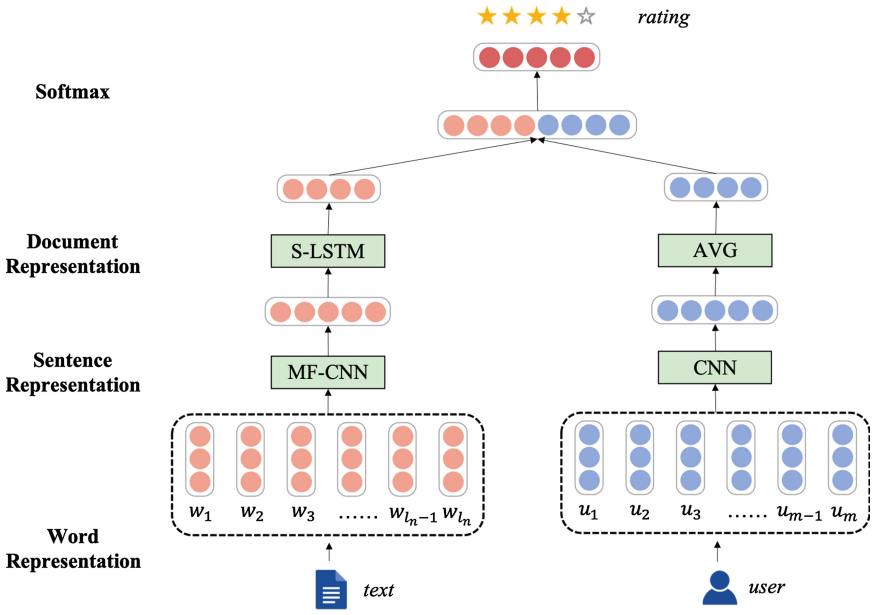


Fig. 3. The architecture of SRPNN model

4.2 Representation Learning for Text Document

We first introduce the left branch of our model which aims at learning the representation of review texts in Fig. 4. Here the first question we need to answer is how to generate the representative vector of a document. The semantics of a document can be obtained from the meanings of its sentences and the rules to compose words into a sentence. Following this routine, we can model the semantics of a document in two steps: we first generate the representation of

a sentence from the word embeddings. Afterwards, we composite the document representation with the embedding matrix of sentences.

For sentence level representation, we adopt the CNN model with multiple *filters* (MF-CNN) to extract the feature vectors from words. In the Word Representation layer, we transform words into representative vectors according to pretrained word embedding. Here $[w_1, w_2, ..., w_{l_n}]$ is the word sequence where l_n is the sentence length. Next we generate local features from the sequence of word embedding using convolution layers. To represent the sentence, we extract unigram, bigram and trigram features from the sentence. We can do it with the *filters* in the convolutional layer: we use multiple convolutional filters with different window sizes as $l_c = 1, 2, 3$ to generate sentence representation. Then the input sequence of the convolution layer is $I_c = [e_i, e_{i+1}, ..., e_{i+l_c-1}]$ $(i \in [1, l_n - l_c + 1], I_c \in \mathbb{R}^{d \times l_c})$. The convolution layer has the following linear transformation:

$$O_c = W_c \cdot I_c + b_c \tag{5}$$

where $W_c \in \mathbb{R}^{l_{oc} \times d \times l_c}$, $b_c \in \mathbb{R}^{l_{oc}}$ are parameters to be learned, l_{oc} denotes the output dimension of linear convolution. Upon the convolution layers, we adopt average pooling operation to aggregate $l_n - l_c + 1$ local features. We also apply *tanh* function to develop the non-linear transformation in the hidden layers. Finally, we obtain three feature vectors with size l_{oc}. We use the average of above three vectors as the feature vector of sentence.

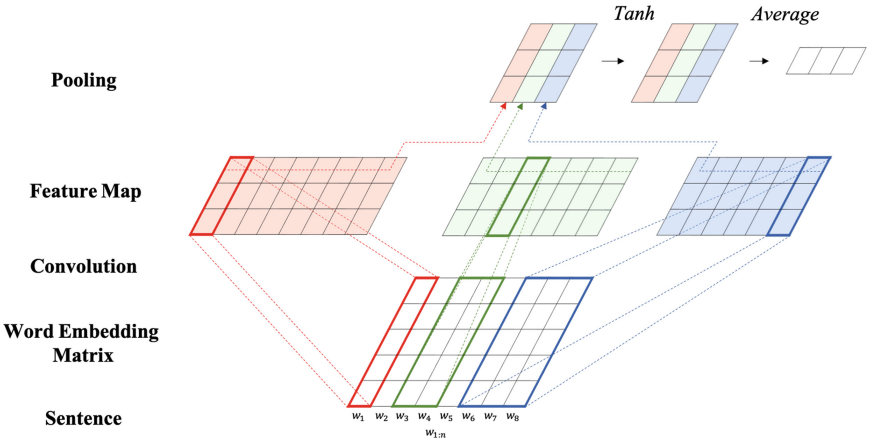


Fig. 4. Learning sentence representation with MF-CNN

To learn the document representation, we adopt the simplified LSTM (S-LSTM) model to generate the sentence-document vector. In this process, we hope to keep as much information of all sentences as possible. So compared with standard LSTM, we discard the output gate and replace the new state C_t with the origin candidate $\tilde{C}_t$ to simplify LSTM, which is similar to the

Gated Recurrent Neural Network (GRNN) model. The gating mechanism of S-LSTM model is shown from Eqs. (6)–(9).

$$i_t = sigmoid(W_i \cdot [s_t; h_{t-1}] + b_i) \tag{6}$$
$$f_t = sigmoid(W_f \cdot [s_t; h_{t-1}] + b_f) \tag{7}$$
$$g_t = tanh(W_g \cdot [s_t; h_{t-1}] + b_g) \tag{8}$$
$$h_t = tanh(i_t \odot g_t + f_t \odot h_{t-1}]) \tag{9}$$

where s_t is the sentence vector at time t, W_i, W_f, W_g are the weight matrices. b_i, b_f, b_g are the offset vectors, $\odot$ denotes the element-wise multiplication, i_t is the input gate, f_t is the output gate, g_t is the new state and h_t is the output at time t. We regard the output of simplified LSTM as the feature vector of the document.

Figure 5 shows the simplified LSTM to learn the document representation from sentences. The input of each time t_i is the sentence vector s_i and the latent output h_{i-1} in the previous time unit; and the output is h_i correspondingly. We get the output of each time unit iteratively and finally obtain h_n as the output of this model.

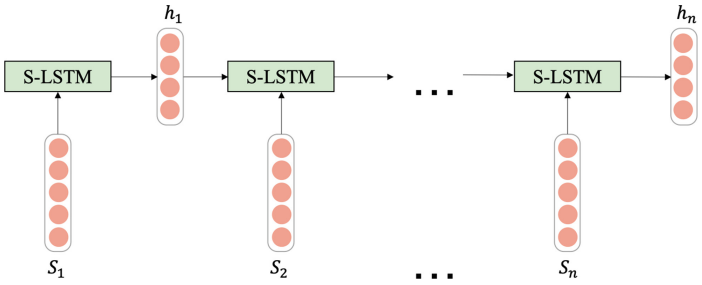


Fig. 5. The simplified LSTM for learning document representation

4.3 Representation Learning for User Document

We then present the right branch of our model to learn the user documents. The representation of user documents is also obtained in two steps, the same as text document: first learning the user sequence and then the document. We generate the user document in the following way: each user in the user network is assigned an identifier. An identifier is regarded as a "word". As we have generated the sequences of users using random walk algorithm, we regard each user sequence as a "sentence". All the user sequences of the same user construct that user's document.

Similar to the text document, we also learn the representation of user document using the CNN model. However, as the sentences in user document are generated through executing the random walk algorithm multiple times, the weights of all the sentences are equivalent. Unlike the text reviews, words in

the user document are not closely related to each other in the contexts. Therefore we only use one window size of filter in convolution layers. We set it as 5 empirically. For learning the document representation, since the sentence order of user documents does not provide semantic information, we do not use LSTM to capture high-level features. Instead we just average the sentence vectors to generate the document representation.

4.4 Output of the Model

As the feature vectors are obtained from both text reviews and user documents, we then generate the joint features by concatenating them, which has been shown in Fig. 3. Then we feed it into a fully connected layer to predict the sentiment label. We adopt softmax function to learn the probability of each classification label. In the training process, we use cross-entropy loss as our loss function. And correspondingly the objective function is denoted in Eq. 10.

$$loss = \sum_{d \in T} \sum_{i=1}^{C} P_i^g(d) \cdot log(P_i(d)) \tag{10}$$

where T is the training set, d is a document in the training set, C is the number of classification categories, $P_i^g(d)$ denotes whether document d belongs to class i (1 when true or 0 when false), $P_i(d)$ is the probability of prediction for document d with class i. We use the Adagrad algorithm to optimize the training process and back-propagation to learn the model parameters. The learning rate is set as 0.03.

Table 1. The statistics of datasets

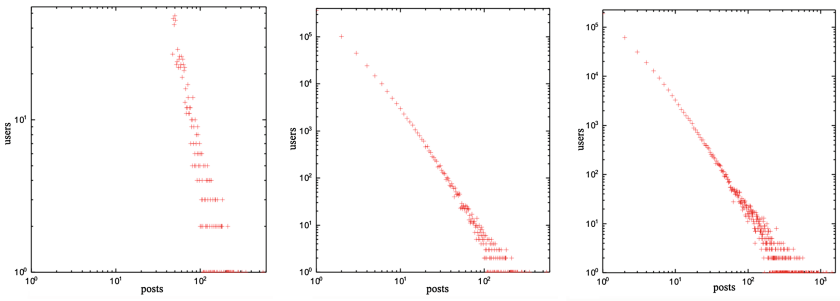
Item	Twitter_1000	Twitter	Yelp
# of posts	76517	1446557	1569264
# of users	1000	596714	366715
# of words	79266	805762	742875
# of posts per user	76.52	2.42	4.28
# of sentence per post	2.86	2.78	9.99
# of words per post	17	16.53	145.02
# of following people per user	8.24	32.97	7.55
sentiment distribution	0.37/0.63	0.5/0.5	0.1/0.09/0.14/0.30/0.37

5 Evaluation

5.1 Data Observations

Datasets. We conduct an extensive set of experiments on two widely used datasets: Twitter and Yelp. Table 1 describes the statistical information of the datasets. Twitter is obtained from the Sentiment140 dataset[1] and Twitter user network [8]. Yelp is obtained from Yelp Dataset Challenge in 2015[2]. The training, validation and test set are constructed by randomly splitting data from a user in the portion of (80/10/10). These two datasets are rather sparse. Following the previous study [33], we also build the dense dataset Twitter_1000 from the top 1000 users with most tweets. The number of classification labels is two for Twitter and five for Yelp respectively.

Figure 6 shows the distribution of post number. We can see that the post number obeys the long-tailed distribution. This can further prove the sparsity of our datasets. We also describe the word frequency and follower number of each datasets in Fig. 7. The distribution of word frequency and follower number can demonstrate the property of text and user respectively. We can see that Twitter and Yelp datasets perform similarly in both word frequency and follower number. Even in Twitter_1000 we have filtered users with highest number of tweets, we can still get the similar distributions. This demonstrates that we can use a unified model to learn the representation of the user and document.



(a) Twitter_1000 (b) Twitter (c) Yelp

Fig. 6. Post distribution of datasets

Experiment Setup. Similar to previous studies [25,33], we use Accuracy and Root Mean Square Error (RMSE) to measure the overall sentiment classification performance.

[1] http://help.sentiment140.com.
[2] https://www.yelp.com/dataset_challenge.

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} 1(\hat{r}_i == r_i) \tag{11}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(\hat{r}_i - r_i)^2}{N}} \tag{12}$$

The RMSE denotes the divergence between predicted sentiment rating ($\hat{r}_i$) and ground truth rating (r_i).

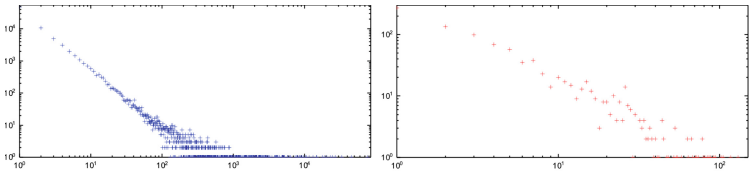
Next we introduce the setting of hyper parameters. For learning the user trust network, we set the random walk step as 10, and every user has 10 random walk sequences. We use the pre-trained word2vec embedding with 200 dimensions. The number of filters of all convolution layers are set as 100. The output dimension of LSTM is set as 60. The dimension of hidden layers in both branches is 50.

5.2 Baseline Methods

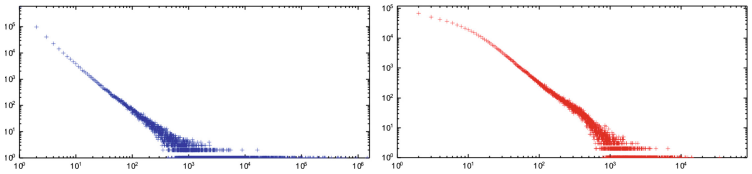
We compare SRPNN with following state-of-the-art baselines of sentiment classification domain, including two feature-based and three deep learning methods:

SVM+N-gram. Following many previous studies [23–25,27,33], we use unigram, bigram and trigram as features to train a SVM classifier as the baseline method.
PMSC. Personalized Microblog Sentiment Classification (PMSC) [33] is a feature-based method assigning personalized weight parameter for each user. As it only supports binary classification, we did not report its results on Yelp dataset.
ParaVec. ParaVec [9] regards each document as a paragraph and learns the paragraph representation. We use the Paravec features and adopt SVM as the classifier to develop a supervised learning.
GRNN. Gated Recurrent Neural Network (GRNN) takes simplified LSTM to classify sentiment polarity by [24]. They study the document modeling methods with deep learning and get significantly better performance than existing approaches.
UPNN. User and Product Neural Network (UPNN) is a personalized sentiment classification model based on convolutional neural network [25].

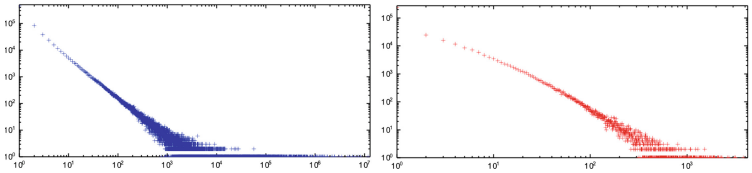
We obtained the source code of above methods from the authors. And for all the methods, we tune the hyper parameters according to original papers and report the best results we achieved.



(a) Twitter_1000



(b) Twitter



(c) Yelp

Fig. 7. Word frequency (left) and follower number (right) of datasets

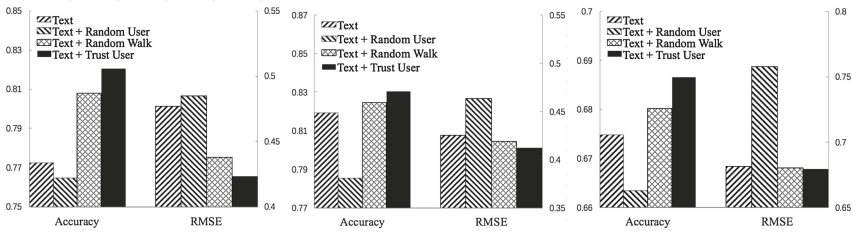
Table 2. Compare with state-of-the-art methods

Model		Twitter_1000		Twitter		Yelp	
		Accuracy	RMSE	Accuracy	RMSE	Accuracy	RMSE
Feature-based methods	SVM+unigram	0.7693	0.4804	0.7883	0.4601	0.6235	0.8356
	SVM+bigram	0.7888	0.4596	0.8067	0.4396	0.6357	0.7762
	SVM+trigram	0.7940	0.4538	0.8169	0.4279	0.6436	0.7580
	PMSC	0.8211	0.4230	–	–	–	–
Deep learning methods	ParaVec	0.7495	0.5005	0.7650	0.4847	0.6016	0.8965
	GRNN	0.7724	0.4770	0.8193	0.4250	0.6748	0.6812
	UPNN	**0.8218**	**0.4222**	0.8192	0.4252	0.6475	0.7576
	SRPNN	0.8205	0.4237	**0.8304**	**0.4118**	**0.6865**	**0.6793**

5.3 Model Comparisons and Analysis

We first compare our SRPNN with state-of-the-art methods. Table 2 shows our experimental results. Note the PMSC method runs out of memory for Twitter datasets, so we cannot report its performance here. We can see that SRPNN outperforms all the other baselines on Twitter and Yelp datasets. The reason is

that when faced with sparse data, other methods fail to capture enough informative features. Although UPNN can take advantage of products information, the benefit is limited due to the problem of data sparsity. The observation that our SRPNN model outperforms GRNN can further demonstrate the effect of user trust network towards personalized sentiment classification: GRNN only focuses on text features with a simplified LSTM, while SRPNN integrates features from both text reviews and user trust network.



(a) Twitter_1000 (b) Twitter (c) Yelp

Fig. 8. Effect of proposed techniques

We then evaluate the effectiveness of our user-trust random walk algorithm by changing the settings of user document. We compared it with 3 baseline methods: *Text* does not include user document; *Text+Random User* uses randomly generated user document; *Text+Random Walk* adopts random walk algorithm on the unweighted directed graph generated from user following relations; *Text+Trust User* is our proposed method. The results are shown in Fig. 8. We can see that *Text+Trust User* achieves the best results. At the same time, *Text+Random Walk* ranks second as it involves user relation information. It is worth noting that *Text+Random User* performs worst. The reason could be that randomly generated user sequence contributes nothing but noise in the user document. It indicates that our user trust network can provide important information of sentiment.

Finally we analyze the result on Twitter_1000. We can see that SRPNN gets very similar results with state-of-the-art methods. This is reasonable because personalized modeling of a user has already been well supported by original data. So there is no data sparsity. As both PMSC and UPNN directly model users, they will have better performance on dense dataset. However, in the real applications such as social media and review rating of products, datasets are often very sparse. Therefore, although SRPNN does not outperform PMSC and UPNN on Twitter_1000, it is still necessary in real-world scenarios.

6 Conclusion

In this paper, we propose Social Relation Powered Neural Network (SRPNN), a deep learning based model for document-level sentiment classification. We propose a random walk algorithm to obtain the sequences of users with user-sentiment consistency so as to generate the user document. We then jointly learn the representation of text and user document. Experimental results on two public datasets demonstrate the effectiveness of our proposed methods.

Acknowledgment. This work was supported by NSFC (91646202), National Key R&D Program of China (SQ2018YFB140235), and the 1000-Talent program.

References

1. Chen, H., Sun, M., Tu, C., Lin, Y., Liu, Z.: Neural sentiment classification with user and product attention. In: EMNLP, pp. 1650–1659 (2016)
2. Dong, L., Wei, F., Zhou, M., Xu, K.: Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In: AAAI, pp. 1537–1543 (2014)
3. Fouss, F., Pirotte, A., Renders, J., Saerens, M.: Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Trans. Knowl. Data Eng. **19**(3), 355–369 (2007)
4. Golbeck, J.: Computing and applying trust in web-based social networks. Ph.D. thesis, University of Maryland, College Park (2005)
5. Hu, X., Tang, L., Tang, J., Liu, H.: Exploiting social relations for sentiment analysis in microblogging. In: WSDM, pp. 537–546 (2013)
6. Jamali, M., Ester, M.: TrustWalker: a random walk model for combining trust-based and item-based recommendation. In: KDD, pp. 397–406 (2009)
7. Kiritchenko, S., Zhu, X., Mohammad, S.M.: Sentiment analysis of short informal texts. J. Artif. Intell. Res. **50**, 723–762 (2014)
8. Kwak, H., Lee, C., Park, H., Moon, S.B.: What is Twitter, a social network or a news media? In: WWW, pp. 591–600 (2010)
9. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: ICML, pp. 1188–1196 (2014)
10. Li, F., Liu, N.N., Jin, H., Zhao, K., Yang, Q., Zhu, X.: Incorporating reviewer and product information for review rating prediction. In: IJCAI, pp. 1820–1825 (2011)
11. Li, Z., Wei, Y., Zhang, Y., Yang, Q.: Hierarchical attention transfer network for cross-domain sentiment classification. In: AAAI, pp. 5852–5859. AAAI Press (2018)
12. Liu, K., Li, W., Guo, M.: Emoticon smoothed language models for Twitter sentiment analysis. In: AAAI (2012)
13. Luo, L., et al.: Beyond polarity: interpretable financial sentiment analysis with hierarchical query-driven attention. In: IJCAI, pp. 4244–4250 (2018)
14. Massa, P., Avesani, P.: Trust-aware recommender systems. In: RecSys, pp. 17–24 (2007)
15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)

16. Mishra, A., Dey, K., Bhattacharyya, P.: Learning cognitive features from gaze data for sentiment and sarcasm classification using convolutional neural network. In: ACL, pp. 377–387 (2017)
17. Mnih, V., Heess, N., Graves, A., Kavukcuoglu, K.: Recurrent models of visual attention. In: NIPS, pp. 2204–2212 (2014)
18. Pang, B., Lee, L.: Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In: ACL (2005)
19. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: KDD, pp. 701–710 (2014)
20. Qian, Q., Huang, M., Lei, J., Zhu, X.: Linguistically regularized LSTM for sentiment classification. In: ACL, pp. 1679–1689 (2017)
21. Qu, L., Ifrim, G., Weikum, G.: The bag-of-opinions method for review rating prediction from sparse text patterns. In: COLING, pp. 913–921 (2010)
22. Socher, R., et al.: Recursive deep models for semantic compositionality over a sentiment treebank. In: EMNLP, pp. 1631–1642 (2013)
23. Song, K., Feng, S., Gao, W., Wang, D., Yu, G., Wong, K.: Personalized sentiment classification based on latent individuality of microblog users. In: IJCAI, pp. 2277–2283 (2015)
24. Tang, D., Qin, B., Liu, T.: Document modeling with gated recurrent neural network for sentiment classification. In: EMNLP, pp. 1422–1432. The Association for Computational Linguistics (2015)
25. Tang, D., Qin, B., Liu, T.: Learning semantic representations of users and products for document level sentiment classification. In: ACL, pp. 1014–1023 (2015)
26. Tang, D., Qin, B., Liu, T.: Aspect level sentiment classification with deep memory network. In: EMNLP, pp. 214–224 (2016)
27. Tang, D., Qin, B., Liu, T., Yang, Y.: User modeling with neural network for review rating prediction. In: IJCAI, pp. 1340–1346 (2015)
28. Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., Qin, B.: Learning sentiment-specific word embedding for Twitter sentiment classification. In: ACL, pp. 1555–1565 (2014)
29. Wang, J., Lin, C., Li, M., Zaniolo, C.: An efficient sliding window approach for approximate entity extraction with synonyms. In: EDBT (2019)
30. Wang, J., Wang, Z., Zhang, D., Yan, J.: Combining knowledge with deep convolutional neural networks for short text classification. In: IJCAI, pp. 2915–2921 (2017)
31. Wang, W., Feng, S., Gao, W., Wang, D., Zhang, Y.: Personalized microblog sentiment classification via adversarial cross-lingual multi-task learning. In: EMNLP, pp. 338–348. Association for Computational Linguistics (2018)
32. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications, vol. 8. Cambridge University Press, Cambridge (1994)
33. Wu, F., Huang, Y.: Personalized microblog sentiment classification via multi-task learning. In: AAAI, pp. 3059–3065 (2016)
34. Wu, Z., Dai, X., Yin, C., Huang, S., Chen, J.: Improving review representations with user attention and product attention for sentiment classification. In: AAAI, pp. 5989–5996. AAAI Press (2018)
35. Zhang, Y., Wu, J., Wang, J., Xing, C.: A transformation-based framework for KNN set similarity search. IEEE Trans. Knowl. Data Eng. (2019)
36. Zhao, K., et al.: Modeling patient visit using electronic medical records for cost profile estimation. In: Pei, J., Manolopoulos, Y., Sadiq, S., Li, J. (eds.) DASFAA 2018. LNCS, vol. 10828, pp. 20–36. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91458-9_2

37. Zhao, Z., Lu, H., Cai, D., He, X., Zhuang, Y.: Microblog sentiment classification via recurrent random walk network learning. In: IJCAI, pp. 3532–3538. ijcai.org (2017)
38. Zhu, L., Galstyan, A., Cheng, J., Lerman, K.: Tripartite graph clustering for dynamic sentiment analysis on social media. In: SIGMOD, pp. 1531–1542 (2014)