International Journal of Database Theory and Application
Vol.6, No.6 (2013), pp.51-62
http://dx.doi.org/10.14257/ijdta.2013.6.6.05

Semi-supervised Sentiment Classification using Ranked Opinion Words

Suke Li and Yanbing Jiang

School of Software and Microelectronics, Peking University

{lisuke, jyb}@ss.pku.edu.cn

Abstract

This work proposes a semi-supervised sentiment classification method which is based on the co-training framework. The proposed method needs to construct three sentiment classifiers. We use common text features to construct the first classifier. We extract opinion words from consumer reviews, and then we ranked these opinion words according to their importance. We also employ extracted opinion words and the ranked co-occurrence opinion words of the extracted opinion words of each review to get the second sentiment classifier. A third sentiment classifier comes into being using non-opinion text features from each review. Based on co-training semi-supervised learning framework, we use the three sentiment classifiers to iteratively get the final sentiment classifier. Experimental results show that our proposed method has better performance than the Self-learning SVM method and the Naive co-training SVM method.

Keywords: *opinion mining, sentiment classification, ranked opinion words*

1. Introduction

With the development of Web 2.0, Web users are used to publishing product comments or reviews in Web logs, Web forums, social networks, and some commercial Web sites after they have finished purchasing activities. These product reviews reflect what products aspects consumers are fond of and what product features consumers dislike. It is believed that product reviews in the Web are subjective, and they are often divided into two categories: positive reviews and negative reviews. Consumers like to express positive sentiment with happy, supportive, approving language, and they can use angry and depressive words to express negative sentiment. Because it is difficult for computers to “understand” natural languages, to do highly accurate sentiment classification is a hard task. So, how to automatically conduct sentiment classification on product reviews has become a research challenge.

Sentiment classification is an important part of opinion mining which is trying to find useful knowledge from Web reviews. Lots of research achievements of opinion mining have been widely applied in business intelligence systems and recommendation systems. Sentiment classification plays a very important role in these applications. Sentiment classification has drawn much research attention in the recent years. Many classification methods have been proposed during the past several years. Some unsupervised methods are easy to be applied and deployed in the actual applications, but these methods could rely on heuristic rules, sentiment lexicons or other language resources, even emphasize on natural language processing techniques such as syntactic and semantic analysis. Although in some cases rule-based methods are very powerful for opinion text mining, machine learning techniques are indeed welcome and popular for sentiment classification. Machine learning-based sentiment classification methods can roughly be classified into two categories: supervised machine learning-based methods and semi-supervised machine learning-based methods. Supervised classification need to label instances to train a sentiment classifier. But most of time it is hard to find useful labeled data, so we must label data by ourselves. The labeling task is time consuming and labor intensive. On the other hand, for every product domain, we could need different training data. Semi-supervised sentiment classification methods only require a small amount of training data and some unlabeled data to get the final sentiment classifier. Our work focuses on semi-supervised sentiment classification which only exploits shallow natural language processing techniques without semantic analysis.

Our method is a kind of semi-supervised sentiment classification method which only needs to label a small number of training instances. The proposed method is based on three observations as follows. The first observation is opinion words with the positive sentiment polarity could have higher possibility to co-occur in positive reviews than in negative reviews, and opinion words with the negative sentiment polarity could have higher possibility to co-occur in negative reviews than in positive reviews. For instance, “great” and “wonderful” have the same positive sentiment polarity, and they could have high possibility in the reviews with positive sentiment instead of negative reviews. The second observation is opinion words in the same review tend to have the same sentiment polarity. The above two observations are not absolute, because in an overall positive review consumer may have negative comment on some product features. Even in a negative review, consumer could express some positive opinions on some product aspects. The third observation is we believe different opinion words can be ranked according to their importance. Some opinion words could have greater contribution to show review holders' sentiment than other opinion words. In other words, opinion words with similar sentiment importance could have similar sentiment strength. So, based on our three observations, we try to propose a research question: whether can we find similar opinion words for the original opinion words in reviews to help improve sentiment classification?

In this work, we firstly extract opinion words from the training data set, unlabeled data set and test data set. Then, we use co-occurrence relationship in both the training data set and unlabeled data set, and we adopt HITS algorithm [1] to rank opinion words. We find opinion words with the similar importance to the original opinion words of each review in the training data set and unlabeled data set. We use uni-grams of the training data to train the first sentiment classifier f_1. We use the original opinion words coming from the training data set and the nearest ranked neighbor opinion words of these original opinion words to build the second sentiment classifier f_2. We use non-opinion text features of the training data set to form the third sentiment classifier f_3. Finally, we employ the co-training framework [2], f_1, f_2, and f_3 to get the final sentiment classifier. Our method is a kind of semi-supervised sentiment classification method which only requires a small number of labeled training instances and some unlabeled instances. Experimental results show that our method has better performance than the Self-learning SVM method and the Naive Co-training SVM method.

2. Related Works

Sentiment classification is one of important tasks of opinion mining. Researchers have made great progress in the field of opinion mining during the past several years. Pang’s survey [3] gives a great summarization for opinion mining and sentiment analysis. Opinion mining has three important tasks including development language resources, sentiment classification and opinion summarization [4]. Some excellent publications focus on word sentiment classification [5, 6], while some early important research works address the problem of document sentiment classification such as [7, 8]. Machine learning techniques were firstly applied in sentiment classification by Pang *et* *al.*, [8]. Pang *et al.*, [8] proved that SVM (support vector machine) had better performance than Naive Bayes and maximum entropy in sentiment classification. Recent year machine learning-based sentiment classification methods are widely studied and seem very promising. Except for various supervised machine learning methods, there are some publications focus on semi-supervised sentiment classification methods, such as [9, 10, and 11]. Recent research work [12] also addressed the problem of semi-supervised sentiment classification by exploiting subjective and objective views of Web reviews. We also compare our work with [12] in this work.

3. Proposed Approach

3.1. Introduction to Our Method

A traditional machine learning-based sentiment classification method needs to manually label a training data set. The label work is usually time-consuming and uninteresting. A semi-supervised machine learning-based sentiment classification method only labels a small number of training instances and some unlabeled instances. The problem is defined as the work [11]. We firstly prepare for a small training data set $T = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\}$, where $\mathbf{x_i}$ is the training instance. Sentiment classification problem can be seen as a binary classification problem, $y_i \in \{-1,+1\}$ is the label of the instance $\mathbf{x_i}$, -1 presents an instance has negative sentiment, and +1 presents it has positive sentiment. We also need an unlabeled data set $U = \{u_1, u_2,..., u_m\}$ which is used in the learning process to get the final sentiment classifier. Let all training instances and unlabeled instances be m dimensional vectors in the real space $X \subseteq R^m$. Then we denote a review as $\mathbf{x_i} = \{x_{i1}, x_{i2}, ..., x_{im}\}$, $\mathbf{x_i}$ is m dimensional input vector. The task of semi-supervised classification problem is to obtain a classification function $f(x)$ to predict the polarity of a consumer review x by exploiting instances in T and U.

$$f : X(T,U) \to L \tag{1}$$

Some opinion words are important for consumers to express strong sentiment. For example, “great” and “wonderful” can be used to show a possibly strong positive sentiment, and “ugly” and “unfriendly” will give much negative sentiment. We have three observations:

- Opinion words with the positive sentiment polarity could have higher possibility to co-occur in positive reviews than in negative reviews; opinion words with the negative sentiment polarity could have higher possibility to co-occur in negative reviews than in positive reviews. For example, “great” and “wonderful” could have higher possibility to co-occur in positive reviews than in negative reviews; “filthy” and “bad” could have higher possibility to be in the same negative review than in the same positive review.
- Opinion words in the same review tend to have the same sentiment polarity. We must say that it is possible that both positive and negative opinion words could exist in the same review. In this case, we assume each review has its major sentiment polarity with major opinion words.
- Different opinion words can have different sentiment strength. These opinion words can show the overall sentiment of reviews. We could find a way to rank opinion words according to their importance.

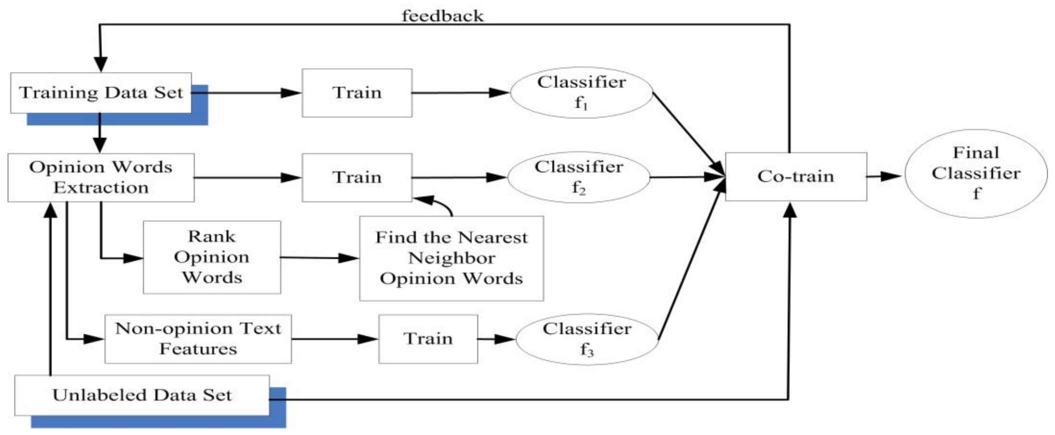

Figure 1. Overview of Our Proposed Method

The research problem is whether we can use these observations to improve sentiment classification. In our approach, we extract opinion words from training and unlabeled data set. We try to construct a graph using the co-occurrence relationship among these opinion words. Then we rank these opinion words using HITS algorithm [1]. Our proposed method has several basic steps, as Figure 1 shows, and the overall algorithm is showed as follows.

- We firstly extract opinion words from training and unlabeled data sets.
- We use the co-occurrence of these extracted words to construct a double-directed graph.
- Using HITS algorithm, we rank opinion words according to their co-occurrence relationship.
- According to the ranking results, we find the nearest neighbor opinion word for every extracted opinion word of each review.
- We train three sentiment classifiers: we get the first sentiment classifier through common uni-grams features. The second classifier is trained using original opinion words and the nearest ranked neighbor opinion words of the original opinion words extracted from each review. The third sentiment classifier is trained using the non-opinion text features.
- At last, we employ co-training [2] framework to get the final sentiment classifier. The co-training method is a bootstrapping method which iteratively put unlabeled instances into training data set.

3.2. Opinion Word Extraction

To simplify the research problem, we only extract adjectives from consumer reviews. Adjectives usually have sentiment polarity, because consumers are used to expressing opinions and describing product features using some adjectives. For example, in the sentence “The staff is helpful and friendly” there are two adjectives (helpful and friendly) with positive sentiment. Some opinion words have negative indicators in the context of negation. In this case, to be simple, a negative indicator and an opinion word together are looked as a single opinion word. For example, in the phrase “not bad”, the word “bad” is an opinion word, and “not” is a negative indicator, and “not bad” is looked as a single opinion word. We extract adjectives with POS (Part-of-Speech) labels of JJ, JJR, and JJS. At the same time, If there has a negative indicator in the context window [-3, 0] of an opinion word, then the negative indicator combined with the sentiment word together constitute a sentiment unit. The context of a word window [-3, 0] means the left distance coverage of the word in a clause is 3 words from the opinion word. These negative indicators including “not”, “no”, “donot”, “do not”, “didn’t”, “did not”, “was not”, “wasn't”, “isnt”, “isn't”, “weren't”, “werent”, “doesn’t”, “doesn't”, “hardly”, “never”, “neither”, and “nor”.

3.3. Co-occurrence Graph Construction and Opinion Word Ranking

We can generate a double-directed graph by exploiting all reviews in the labeled data set and the unlabeled data set. Co-occurrence of two opinion words means they must appear in the same consumer review. In the graph, opinion words are used as graph nodes, and the edges present co-occurrence relationship. For each co-occurrence opinion word pair, a double direction edge is generated. When we finish generating the graph, we can use some graph ranking method to rank these opinion words. Suppose the directed graph is denoted as $G = (V, E)$, while V is the node set, and E is the edge set. Let M be the adjacency matrix of the graph, if opinion word i co-occurs with opinion word j, then $M_{ij} = 1$ and $M_{ji} = 1$; otherwise, $M_{ij} = 0$ and $M_{ji} = 0$ (Graph G is a double-directed graph).

We use HITS algorithm to do the task of ranking. HITS algorithm is famous for its distinguished performance in link analysis for ranking Web pages. Some research work also adopts HITS to rank product features [13]. We believe HITS can also be used to rank opinion words. In order to use HITS algorithm, we need to computes two kinds of scores for each Web page. One is the hub score, the other is the authority score. We assume that for each opinion word in our co-occurrence graph h_i is the hub score of the opinion word i, and a_i is the authority score of the opinion word i. The two kinds of scores can be iteratively computed using Equation (2) and Equation (3). The authority score of an opinion word is the sum of hub scores of opinion words that links to the opinion word, and the hub score of an opinion word is the sum of authority scores of opinion words that the opinion word points to. The mutual reinforcement relationship of authorities and hubs is the key philosophy in the HITS algorithm.

$$a_i = \sum_{(j,i)\in E} h_j \tag{2}$$

$$h_i = \sum_{(i,j)\in E} a_j \tag{3}$$

If our opinion word number is n, then we present the authority scores of all opinion words as a column vector $\boldsymbol{A} = (a_1, a_2, ..., a_n)^T$. We denote $\boldsymbol{H} = (h_1, h_2, ..., h_n)^T$ as the column vector with the hub scores. The idea is we iteratively compute the authority score vector and the hub score vector using Equation (4) and Equation (5) until the algorithm converges. We set all the elements of $\boldsymbol{A}$ and $\boldsymbol{H}$ the same value of 1’s. The initial vector for $\boldsymbol{A}$ and $\boldsymbol{H}$ is $\boldsymbol{A} = (1,...,1)^{\boldsymbol{T}}$, $\boldsymbol{H} = (1,...,1)^{\boldsymbol{T}}$.

$$A = M^T B \tag{4}$$

$$H = MA \tag{5}$$

3.4. Using Ranked Opinion Words and Co-training Framework to Conduct Sentiment Classification

The co-training method [2] is typical semi-supervised self-boost algorithm framework. The co-training method uses different views to train the final classifier. We generate three sentiment classifiers which will be used to get the final sentiment classifier in our co-training framework. The first sentiment classifier takes the frequencies of uni-grams extracted from the training data set as training features. In this case, we remove the tokens whose sizes are less than 3, as well as some stop words. After the preprocessing action, we use ***TinySVM*** [14] to get the first sentiment classifier f_1.

The second sentiment classifier is trained by extracted opinion words which have POS tags of “JJ”, “JJR”, and “JJS”. If in the context window of an opinion word there is a negative indicator, then the negative indicator as well as the opinion word together to form an opinion unit which is looked as a single training feature. Negative indicators include “not”, “no”, “donot”, “don't”, “didn’t”, “wasn’t”, “isn't”, “isnt”, “weren't”, “werent”, “doesn't”, “doesn’t”, “didn't”, “hardly”, “never”, “neither” and “nor”. We use frequencies of these extracted opinion words or sentiment units from training data set to get the second sentiment classifier f_2.

In order to get the third sentiment classifier, we must get the nearest neighbor opinion word of every original opinion word of each review. Suppose O_T is the set of total opinion words extracted from the training data set. Let O_U be the set of total opinion words extracted from the unlabeled data set. Let $O_{TU} = O_T \cup O_U$, for each review r_i we will get an opinion word set O_i for r_i, while $O_i \subseteq O_{TU}$. O_i could include one or several opinion words. According to the adjacent matrix M, we can get the co-occurrence opinion word set for every element of O_i. For any element o_{ij}, $o_{ij} \in O_i$, $0 \leq j < |O_i|$, we denote o_{ij}’s co-occurrence opinion word set as N_{ij}. We select the nearest neighbor opinion word $n_{ijz} \in N_{ij}$ which has the nearest authority score to o_{ij}, here $n_{ijz} \in N_{ij}$, $n_{ijz} \neq o_{ij}$, $0 \leq z < |N_{ij}|$, satisfying the condition

$$z = \arg\min_z |A(n_{ijz}) - A(o_{ij})|, \tag{6}$$

where $A(n_{ijz})$ and $A(o_{ij})$ are the authority scores in the HITS algorithm described in Section 3.3. Using Equation (6), we put the selected neighbor opinion word n_{ijz} into r_i's neighbor opinion word set N_i. Finally, we get the neighbor opinion word set N_i for O_i, where $|N_i| \leq |O_i|$. (Note: if an opinion word has no any co-occurrence opinion word, then we could get $|N_i| < |O_i|$.) The detailed steps is shown as Algorithm 1.

```
Algorithm 1: Sentiment Classification Algorithm Based on Ranked Opinion Words

Input: Training data set $T=\{t_1, t_2,...,t_x\}$, $T$ includes balanced positive reviews and negative
       reviews; Unlabeled data set $U=\{u_1, u_2, ..., u_y\}$;

Output: Sentiment classifier $C$;

BEGIN

1. Extract opinion words from $T$ and $U$, we get the training opinion word set $O_T$ and the
   unlabeled opinion word set $O_U$.

2. Construct a graph and get adjacent matrix $M$ using co-occurrence relationship (if two
   opinion words are in the same review) among opinion words coming from $O_T$ and $O_U$
   ;

3. $O_{TU} = O_T \cup O_U$ , rank the opinion words of $O_{TU}$ using HITS algorithm;
```

```
4.  For each training instance in training data set $T$ and unlabeled data set $T$, we get the
    nearest neighbor opinion word set $N_i$ of its original opinion word $O_i$ set using
    Equation (6), where $|N_i| \leq |O_i|$;
5. WHILE {There are unlabeled instances left in the unlabeled data set}
6.      Use training data set $T$ and SVM to get the first sentiment classifier $f_1$;
7.      Use original opinion words extracted from the training instances together with the
        nearest neighbors of extracted original opinion words to train SVM to get the
        second classifier $f_2$;
8.      Use the non-opinion text features to get the third sentiment classifier $f_3$;
9.      Use $f_1$ to classifier instances in $U$, and get positive instance set $P_{f1}$ ~ and Negative
        instance set $N_{f1}$ (In each iteration we take the most 50 possible classified
        instances);
10.     Use $f_2$ to classify unlabeled data set $U$, and get positive instance set $P_{f2}$ and
        negative instance set $N_{f2}$ (In each iteration we take the most 50 possible classified
        instances);
11.     Use $f_3$ to classify unlabeled data set $U$, and get positive instance set $P_{f3}$ and
        negative instance set $N_{f3}$ (In each iteration we take the most 50 possible
        classified instances);
12.     $L=L \cup P_{f1} \cup N_{f1} \cup P_{f2} \cup N_{f2} \cup P_{f3} \cup N_{f3}$;
13. ENDWHILE;
14. $C=f_1$;
15. RETURN C;
END.
```

4. Experiments

4.1. Experimental Data

We crawled mobile phone and laptop reviews from ***Amzaon*** [15] and hotel reviews from ***Tripadvisor*** [16] respectively. We used ***OpenNLP*** [17] get sentences and POS tags for these reviews. These data sets were also used in the previous publications [11, 12]. Statistics on the experimental data sets are shown in Table 1. For example, the hotel data set has 1000 positive reviews and 1000 negative reviews, and these reviews are segmented into 18694 sentences. The sentiment polarity of a review in a training data set is assigned according to their review ratings. A review rating is a real number ranging from 1 to 5. When the a review rating is greater than 3, then the review is a positive review; when a review rating is less than 3, the review is a negative review.

Table 1. Experimental Statistics

Data	Positive reviews #	Negative reviews #	Sentences #
phone	1000	1000	28811
laptop	1000	1000	14814
hotel	1000	1000	18694

4.2. Compared Methods

4.2.1 Self-learning SVM Method: The earliest use of SVM (Support Vector Machine) [18] to determine the sentiment polarity of consumer reviews is Pang's research work [8]. It is believed SVM is one of the best text classification methods. Because our proposed method is semi-supervised sentiment classification method, we try to compare our method with the Self-learning SVM method. Therefore, the Self-learning SVM method is our baseline. We use the frequencies of uni-grams of reviews as the training and test features for SVM. We employ ***TinySVM*** [14] software to conduct SVM-based sentiment classification.

```
Algorithm 2: Self-learning SVM Method [11, 12]
Input: Training set $T = \{t_1, t_2, ..., t_x\}$ , $T$ includes positive reviews and negative reviews;
       Unlabeled data set $U=\{u_1, u_2, ..., u_y\}$ ;
Output: Sentiment classifier $C$;
BEGIN
1. $n$ is the number of selected reviews which are the most likely correctly classified reviews;
2. Use SVM to get the initial sentiment classifier $C$ on training data set $T$;
3. WHILE {the unlabeled data set is not empty}
4.     Use sentiment classifier $C$ to classify the unlabeled instances in $U$: get positive set $P$ and the negative set $N$;
5.     If $|P|> = d$, then select the most likely correctly classified $d$ instances from $P$ (the set is $P_d$ ) into $T$, $T=T \cup P_d$, $P=P-P_d$; otherwise put all the instances in the $P$ into $T$, $T = T \cup P$ ;
6.     If $|N|> = d$ , then select the most likely correctly classified $d$ instances from $N$ (the set is $N_d$ ) into $T$ , $T=T \cup N_d$ , $N=N-N_d$ ; otherwise put all the instances in the $N$ into $T$, $T= T \cup N$ ;
7.     Employ SVM to train a new sentiment classifier $C$ on the current training data set $T$ ;
8. ENDWHILE;
9. RETURN $C$;
END.
```

Self-learning SVM is a bootstrap approach to learning as Algorithm 2 shows. The method is also used as the baseline in the previous works [11, 12]. The algorithm iteratively selects the most likely correctly classified reviews which are determined by their distances to classification hyperplane, and put them into the training data set. A new sentiment classifier is built by training on the new training data set. Repeat the above steps until there are no unlabeled review that can be added to the training data set so far. When a classified review has greater distance from SVM hyperplane, the review is considered have higher probability to be correctly classified.

4.2.2 Naive Co-training SVM Method: The Naive Co-training SVM method is based on the same co-training framework as Algorithm 3 which has been described as the Two-view Co-training Method in the first author's previous work [12]. The only difference between Algorithm 1 and Algorithm 3 is that Algorithm 1 uses the HITS algorithm to rank opinion words, and select the nearest neighbor opinion words according to ranking information of opinion words to train the sentiment classifier f_2, while in the Naive Co-Training SVM sentiment classification method, we only use the original opinion words to train the sentiment classifier f_2.

Algorithm 3: Naive Co-training SVM Method (Two-view Co-training Method [12])

```
Input: Training data set $T=\{t_1, t_2, ..., t_x\}$, $T$ is balanced data set; Unlabeled data set
       $U=\{u_1, u_2, ..., u_y\}$;
Output: Sentiment classifier $C$ ;
BEGIN
1. WHILE {There are unlabeled instances left in the unlabeled data set}
2.     Use training data set $T$ and SVM to get the first sentiment classifier $f_1$ ;
3.     Use opinion words extracted from the training instances to train SVM to get the
       second classifier $f_2$ ;
4.     Use non-opinion text features from training data set and SVM to get the third
       sentiment classifier $f_3$ ;
5.     Use $f_1$ to classifier instances in $U$, and get positive instance set $P_{f1}$ and Negative
       instance set $N_{f1}$ (each iteration we take the most 50 possible classified instances;
6.     Use $f_2$ to classify unlabeled data set $U$, and get positive instance set $P_{f2}$ and
       negative instance set $N_{f2}$ (each iteration we take the most 50 possible classified
       instances);
7.     Use $f_3$ to classify unlabeled data set $U$ , and get positive instance set $P_{f3}$ and
       negative instance set $N_{f3}$ (each iteration we take the most 50 possible classified
       instances);
8.     $L=L \cup P_{f1} \cup N_{f1} \cup P_{f2} \cup N_{f2} \cup P_{f3} \cup N_{f3}$;
9. ENDWHILE;
10. $C=f_1$ ;
11. RETURN C ;
End.
```

4.3. Experiments

For each category data set, we divided the whole data set into training, unlabeled and test data sets respectively. The test data set of each product is randomly sampled from the product data set. Each test data set includes 800 instances, 400 positive instances and 400 negative instances. For example, if the training instance number is 100 as Table 2 shows, it means we have 50 positive instances and 50 negative instances for training. The remaining 1100 instances are used as unlabeled instances. We use accuracy to evaluate our proposed method, as Equation (7).

$$Accuracy = \frac{\text{number of right classified instances}}{\text{total number of test instances}} \tag{7}$$

Table 2. Experimental Results on the Phone Data Set

Training	**Unlabeled**	**Test**	**Self-learning SVM**	**Naive Co-training SVM**	**Proposed Method**
100	1100	800	66%	**74%**	72.25%
200	1000	800	64%	75.25%	**77.38%**
300	900	800	72.13%	76.63%	**80.25%**
400	800	800	73.38%	78%	**78.88%**

Table 2 gives experimental results of our proposed method on the phone data set. In Table 1, we can see that our proposed method has the best performance, while the Self-learning SVM method has the worst accuracy. Only in the case when the number of training data instance is 100, the Naive Co-training method outperforms the proposed method.

Table 3. Experimental Results on the Laptop Data Set

Training	**Unlabeled**	**Test**	**Self-learning SVM**	**Naive Co-training SVM**	**Proposed Method**
100	1100	800	64.63%	72.38%	**74.00%**
200	1000	800	70.50%	**76.38%**	**76.38%**
300	900	800	67.50%	72.75%	**77.50%**
400	800	800	69.13%	**78.13%**	77.63%

Table 3 shows that both our proposed method and the Naive Co-training method have better performance than the Self-learning SVM method on the laptop data set. But in this case, the proposed method and the Naive Co-training SVM method are well-matched, and the proposed method has little advantage than the Naive Co-training method.

Table 4. Experimental Results on the Hotel Data Set

Training	Unlabeled	Test	Self-learning SVM	Naive Co-training SVM	Proposed Method
100	1100	800	68.88%	79.63%	**80.50%**
200	1000	800	76.25%	83.25%	**83.50%**
300	900	800	79.50%	82.63%	**83.38%**
400	800	800	79.75%	**84.50%**	85.25%

In Table 4, we can see our proposed method has the best performance among the three methods. In this case, ranked opinion words are very helpful for the sentiment classification. The Naive Co-Training SVM method seems stable and it has better performance than the Self-learning SVM method. We can see the classification accuracy will improve if we give more training instances. In summary, our proposed method is effective on the empirical data sets.

5. Conclusion

This work focuses on semi-supervised sentiment classification. We proposed a semi-supervised sentiment classification method which focuses on using ranked opinion words to build a semi-supervised sentiment classifier based on the co-training framework. The method itself doesn't rely on other language resources. Our method only needs a small number of labeled training instances and some unlabeled instances. We find that ranked opinion words are helpful for improving the final sentiment classification accuracy. Experimental results on empirical data sets show our method is effective and promising, and it outperforms the Self-learning SVM method and the Naive Co-training SVM Method. In the future, we will continue our research to find more effective sentiment classification methods.

Acknowledgements

We thank anonymous reviewers for their constructive comments. The paper is supported by the National Natural Science Foundation of China under Grant 61170002.

References

[1] J. Kleinberg, "Authoritative sources in a hyperlinked environment", Journal of ACM, vol. 46, no.5, **(1999)**, pp. 604-632.
[2] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training", Proceedings of the 11th Annual Conference on Computational Learning Theory", **(1998)**, pp. 92-100.
[3] B. Pang and L. Lee, "Opinion mining and sentiment analysis", Foundations and Trends in Information Retrieval, vol. 2, no. 1-2, **(2008)**.
[4] D. Lee, O. Jeong and S. Lee, "Opinion mining of customer feedback data on the web", Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication, **(2008)**, pp. 230-235.
[5] P. Turney and M. L. Littman, "Measuring praise and criticism: Inference of semantic orientation from association", ACM Transaction on Information System, vol. 21, no. 4, **(2003)**, pp. 315-346.
[6] V. Hatzivassiloglou and K. McKeown, "Predicting the semantic orientation of adjectives", Proceedings of the 8th Conference on European Chapter of the Association for Computational Linguistics, **(1997)**, pp. 174-181.

[7] P. Turney, "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews", Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, **(2002)**, pp. 417-424.
[8] B. Pang, L. Lee and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques", Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, **(2002)**, pp. 79-86.
[9] S. Zhou, Q. Chen and X. Wang, "Active deep networks for semi-supervised sentiment classification", Proceedings of the 23rd International Conference on Computational Linguistics: Posters, **(2010)**, pp. 1515-1523.
[10] S. Li, C. R. Huang, G. Zhou and S. Y. M. Lee, "Employing personal/impersonal views in supervised and semi-supervised sentiment classification", Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, **(2010)**, pp. 414-423.
[11] S. Li and J. Hao, "Spectral clustering-based semi-supervised sentiment classification", Proceedings of 8th Advanced Data Mining and Applications, Lecture Notes in Computer Science, Springer, vol. 7713, **(2012)**, pp. 271-283.
[12] S. Li, "Sentiment classification using subjective and objective Views", International Journal of Computer Applications, vol. 80, no. 7, **(2013)**, pp. 30-34.
[13] L. Zhang, B. Liu, S. Lim and E. O'Brien-Strain, "Extracting and ranking product features in opinion documents", Proceedings of the 23rd International Conference on Computational Linguistics: Posters, **(2010)**, pp. 1462-1470.
[14] http://chasen.org/~taku/software/TinySVM/.
[15] http://www.amazon.com.
[16] http://www.tripadvisor.com.
[17] http://opennlp.apache.org.
[18] C. Cortes and V. Vapnik, "Support-vector networks", Machine Learning, vol. 20, no. 3, **(1995)**, pp. 273-297.

Authors

Suke Li, he received his Ph.D. degree in computer science from Peking University (2012). He is currently an assistant professor in Peking University, China. His research interests include financial data mining, Web mining and retrieval, opinion mining, social networks, and information security

Yanbing Jiang, he received his Ph.D. degree in computer science from Peking University (2004). Now he is an associate professor of Peking University. His research interests include software development methodology, object-oriented software development technology, model-driven software development technology, "Cloud + End" mobile Internet software development and reconstruction technology.