

## SentiRank: Cross-Domain Graph Ranking for Sentiment Classification

Qiong Wu<sup>1,2</sup>, Songbo Tan<sup>1</sup>, Haijun Zhai<sup>3</sup>, Gang Zhang<sup>1</sup>, Miyi Duan<sup>1</sup> and Xueqi Cheng<sup>1</sup>

<sup>1</sup>Institute of Computing Technology, Chinese Academy of Sciences, China

<sup>2</sup>Graduate University of Chinese Academy of Sciences, China

<sup>3</sup>University of Science & Technology of China, China

{wuqiong,tansongbo,zhaihaijun}@software.ict.ac.cn

### Abstract

*Sentiment classification is attracting more and more attention because of its great benefits to social and human life. Usually supervised classification approaches perform well in sentiment classification, but the performance decreases sharply when transferred from one domain to another domain. In this paper, we propose an approach, SentiRank, which integrates the sentiment orientations of the documents into the graph-ranking algorithm for cross-domain sentiment classification. We apply the graph-ranking algorithm using the accurate labels of old-domain documents as well as the “pseudo” labels of new-domain documents, and investigate their relative importance for cross-domain sentiment classification. The experiment results indicate that the proposed algorithm could improve the performance of cross-domain sentiment classification dramatically.*

### 1. Introduction

Sentiment classification [1, 2] is a special kind of text classification where a document is classified as positive or negative for an object (i.e. book, hotel, etc.). With the rapid growth of web pages, there is much sentiment information about a given object for human beings. Sentiment classification helps to analyze, organize and summarize the opinions of the huge amount of web pages quickly and easily.

In most cases, supervised classification methods can perform well in sentiment classification. Pang et al. [2] conducted experiments to show that standard machine learning techniques definitively outperform human-produced baselines. But supervised learning needs two conditions to guarantee the accuracy of classification. First, training data should be enough and labeled well so that the model can be trained sufficiently. Second, training data and test data should have the same distribution so that test data can share the information got from training data. But the two conditions usually can't be satisfied in practice. For the first condition, labeling data involves much human labor and is time-consuming, so it's often hard to get high-quality labeled data for a

new domain; for the second condition, the existing labeled data and test data are often from related but different domains between which there often exists a considerable distribution difference. This is because different domains tend to use different domain-specific words to express sentiment. For example, the word “portable” may be positive in electronics reviews, but it means nothing in hotel reviews. This raises a research field called cross-domain sentiment classification.

Cross-domain sentiment classification (that is, sentiment-transfer) is a new study field. In recent years, only a few works are about this new and significant field. They are generally divided into two categories. The first one needs a small amount of labeled training data for the new domain (e.g. [4]). The second one needs no labeled data for the new domain, and the works of Blitzer et al. [5] and Tan et al. [6, 7] are the representative ones. In this paper, we concentrate on the second category which proves to be used more widely.

Graph-ranking algorithm (e.g. PageRank [11]) has been successfully used in many fields whose idea is to give a node high score if it is strongly linked with other high-score nodes. In this study, we propose an approach called *SentiRank*. It extends the graph-ranking algorithm for sentiment-transfer by integrating the sentiment orientations of the documents, which could be considered as a sentiment-transfer version of the graph-ranking algorithm. We also investigate the relative importance of the old-domain information and the new-domain information for sentiment classification. In this algorithm, we assign a score for every unlabelled document to denote its extent to “negative” or “positive”, then we iteratively calculate the score making use of the accurate labels of old-domain data as well as the “pseudo” labels of new-domain data, and the final score for sentiment classification is achieved when the algorithm is converged, so we can label the new-domain data basing on these scores.

In order to evaluate the effectiveness and robustness of *SentiRank*, we conduct experiments on three domain-specific sentiment data sets. The experiment results show that *SentiRank* can dramatically improve the accuracy when transferred to a new domain. And we also

experiment to test the parameters sensitivity. The results show that our algorithm is not sensitive to these parameters.

## 2. Related Work

### 2.1 Supervised Sentiment Classification

Supervised sentiment classification (e.g. [2, 8]) is to determine the sentiment of texts. It is a special kind of classification that classifies texts according to their opinions or attitudes (negative or positive, happy or sad et al.) of a given subject. It has drawn more and more attention because of its applications in many aspects. Pang et al. [2] applied three traditional supervised classification methods to sentiment classification, and the experiments showed that standard machine learning techniques definitively outperform human-produced baselines. Cui et al. [8] presented experiments with different machine-learning algorithms using huge amount of online product reviews. The experimental results showed that a discriminating classifier combined with high order n-grams as features could achieve better performance.

However, supervised sentiment classification requires that labeled and unlabeled data should be under the same distribution, so that the classifier built by the labeled data could be well applied to the unlabeled data. But in our cross-domain classification field, the labeled and unlabeled data are from different domains, and often have different distributions. This is inconsistent with the basic requirement of supervised classification, and this kind of effective methods can't be directly used in cross-domain sentiment classification.

### 2.2 Transfer Learning

Transfer learning aims to utilize data from other domains or time periods to help current learning task. Recently, transfer learning has drawn more and more attention as an important research field in machine learning. Many researchers have been working on this field and have applied many approaches (e.g. [9, 10]). DaumeIII and Marcu [9] studied the domain-transfer field in statistical natural language processing using a specific Gaussian model. Jiang and Zhai [10] presented a two-stage approach for transfer learning. At the first generalization stage, they got a set of features generalizable across domains, and at the second adaptation stage, they picked up useful features specific to the target domain.

Different from these works, we design algorithms for transfer learning in the context of sentiment classification.

## 3. The Proposed Approach

### 3.1 Problem Definition

In this paper, we have two document sets: the test data  $D^U = \{d_1, \dots, d_n\}$  where  $d_i$  is the term vector of the  $i^{\text{th}}$  text document and each  $d_i \in D^U (i = 1, \dots, n)$  is unlabeled; the training data  $D^L = \{d_{n+1}, \dots, d_{n+m}\}$  where  $d_j$  represents the term vector of the  $j^{\text{th}}$  text document and each  $d_j \in D^L (j = n+1, \dots, n+m)$  should have a label from a category set  $C = \{\text{negative}, \text{positive}\}$ . We assume the training dataset  $D^L$  is from the related but different domain with the test dataset  $D^U$ . Our objective is to maximize the accuracy of assigning a label in  $C$  to  $d_i \in D^U (i = 1, \dots, n)$  utilizing the training data  $D^L$  in another domain.

### 3.2 Overview

The proposed algorithm is based on the following presumptions:

(1) Let  $W^L$  denote the word space of old domain,  $W^U$  denote the word space of new domain.  $W^L \cap W^U \neq \Phi$ .

(2) The labels of documents appear both in the training data and the test data should be the same.

In our algorithm, we initialize every document a score ("1" denotes positive, and "-1" denotes negative) to represent its degree of sentiment orientation, and we call it sentiment score.

Based on graph-ranking algorithm, it is thought that if a document is strongly linked with positive (negative) documents, it is probably positive (negative). And this is the basic idea of learning from a document's neighbors.

Our algorithm integrates the sentiment orientations of the documents into the graph-ranking algorithm. In more detail, the proposed algorithm considers instances from both training domain and test domain:

Training Domain: Training data are labeled beforehand, so we can score the test data utilizing the correct labels of training data.

Test Domain: Test data are similar to each other, so we can score the test data utilizing their "pseudo" labels of each other.

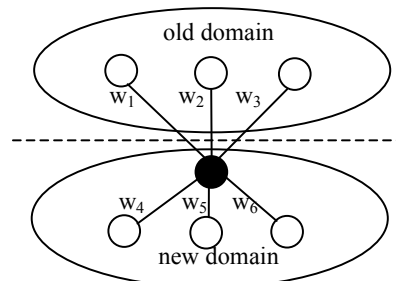


Figure 1. SentiRank Algorithm

Figure 1 illustrates how to score the test data utilizing information of both domains.

The proposed algorithm calculates the sentiment score of every unlabelled document by learning from its neighbors in both old domain and new domain, and then iteratively calculates the scores with a unified formula. Finally, the algorithm converges and each document gets its sentiment score. When its sentiment score is between 0 and 1, the document should be classified as “positive”. The closer its sentiment score is near 1, the higher the “positive” degree is. Otherwise, when its sentiment score is between 0 and -1, the document should be classified as “negative”. The closer its sentiment score is near -1, the higher the “negative” degree is.

### 3.3 Score Documents

#### Score Documents Using Old-domain Information

This section introduces how to calculate the sentiment score of  $D^U$  by making use of labels of  $D^L$ .

We build a graph whose nodes denote documents in both  $D^L$  and  $D^U$  and edges denote the content similarities between documents. If the content similarity between two documents is 0, there is no edge between the two nodes. Otherwise, there is an edge between the two nodes whose weight is the content similarity. The content similarity between two documents is computed with the cosine measure. We use an adjacency matrix  $U$  to denote the similarity matrix between  $D^U$  and  $D^L$ .  $U=[U_{ij}]_{n \times m}$  is defined as follows:

$$U_{ij} = \frac{d_i \bullet d_j}{\|d_i\| \times \|d_j\|}, \quad i=1, \dots, n, j=n+1, \dots, n+m \quad (1)$$

The weight associated with term  $t$  is computed with  $tf_i df_j$ , where  $tf_i$  is the frequency of term  $t$  in the document and  $idf_j$  is the inverse document frequency of term  $t$ , i.e.  $1+\log(N/n_t)$ , where  $N$  is the total number of documents and  $n_t$  is the number of documents containing term  $t$  in a data set.

In consideration of convergence, we normalize  $U$  to  $\hat{U}$  by making the sum of each row equal to 1:

$$\hat{U}_{ij} = \begin{cases} U_{ij} / \sum_{j=1}^m U_{ij}, & \text{if } \sum_{j=1}^m U_{ij} \neq 0 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

In order to find the neighbors (in another word, the nearest documents) of a document, we sort every row of  $\hat{U}$  to  $\tilde{U}$  in descending order. That is:  $\tilde{U}_{ij} \geq \tilde{U}_{ik}$  ( $i=1, \dots, n; j, k=1, \dots, m; k \geq j$ ).

Then for  $d_i \in D^U$  ( $i=1, \dots, n$ ),  $\tilde{U}_{ij}$  ( $j=1, \dots, K$ ) corresponds to  $K$  neighbors in  $D^L$ . So we can get its  $K$

neighbors. We use a matrix  $N=[N_{ij}]_{n \times K}$  to denote the neighbors of  $D^U$  in old domain, with  $N_{ij}$  corresponding to the  $j^{\text{th}}$  nearest neighbor of  $d_i$ .

At last, we can calculate  $s_i$  ( $i=1, \dots, n$ ) using the sentiment scores of the  $d_i$ 's neighbors as follows:

$$s_i^{(k)} = \sum_{j \in N_{i \bullet}} (\hat{U}_{ij} \times s_j^{(k-1)}), \quad i=1, \dots, n \quad (3)$$

where  $i \bullet$  means the  $i^{\text{th}}$  row of a matrix, and  $s_i^{(k)}$  denotes the  $s_i$  at the  $k^{\text{th}}$  iteration.

#### Score Documents Using New-domain Information

This section introduces how to calculate the sentiment score of  $D^U$  by making use of  $D^U$  itself.

Similarly, a graph is built, in which each node corresponds to a document in  $D^U$  and the weight of the edge between any different documents is computed by the cosine measure. We use an adjacency matrix  $V=[V_{ij}]_{n \times n}$  to describe the similarity matrix. And  $V$  is similarly normalized to  $\hat{V}$  to make the sum of each row equal to 1. Then we sort every row of  $\hat{V}$  to  $\tilde{V}$  in descending order, thus we can get  $K$  neighbors of  $d_i \in D^U$  ( $i=1, \dots, n$ ) from  $\tilde{V}_{ij}$  ( $j=1, \dots, K$ ), and we use a matrix  $M=[M_{ij}]_{n \times K}$  to denote the neighbors of  $D^U$  in the new domain. Finally, we can calculate  $s_i$  ( $i=1, \dots, n$ ) using the scores of the  $d_i$ 's neighbors as follows:

$$s_i^{(k)} = \sum_{j \in M_{i \bullet}} (\hat{V}_{ij} \times s_j^{(k-1)}), \quad i=1, \dots, n \quad (4)$$

### 3.4 SentiRank Algorithm

#### Initialization

In order to initialize the sentiment score vector  $S^{(0)}$ , we need to classify the test data to get their initial labels using a traditional classifier. There are many kinds of classifiers. For simplicity, we take prototype classification algorithm [3] as an example here.

Firstly, we compute the center vector  $C_{neg}$  and  $C_{pos}$  for negative document set ( $D_{neg}^L$ ) and positive document set ( $D_{pos}^L$ ) of  $D^L$  as follows:

$$C_l = \sum_{i \in D_l^L} d_i / |D_l^L|, \quad l \in \{neg, pos\} \quad (5)$$

where  $|\cdot|$  is the cardinality of a set.

Secondly, we calculate the similarity between  $d_i \in D^U$  ( $i=1, \dots, n$ ) and each center vector with the cosine measure as follows:

$$Sim_l = \frac{d_i \cdot C_l}{\|d_i\| \times \|C_l\|}, \quad l \in \{neg, pos\} \quad (6)$$

Thirdly, we assign  $d_i \in D^U$  ( $i = 1, \dots, n$ ) a class label corresponding to the more similar center vector.

Fourthly, we give “-1” to  $s_i$  if  $d_i$ 's label is “negative”, and “1” if “positive”.

Finally,  $s_i^{(0)}$  ( $i = 1, \dots, n$ ) is normalized as follows to make the sum of positive scores of  $D^U$  equal to 1, and the sum of negative scores of  $D^U$  equal to -1:

$$s_i^{(0)} = \begin{cases} s_i^{(0)} / \sum_{j \in D_{neg}^U} (-s_j^{(0)}), & \text{if } s_i^{(0)} < 0 \\ s_i^{(0)} / \sum_{j \in D_{pos}^U} s_j^{(0)}, & \text{if } s_i^{(0)} > 0 \end{cases} \quad i = 1, \dots, n \quad (7)$$

where  $D_{neg}^U$  and  $D_{pos}^U$  denote the negative and positive document set of  $D^U$  respectively. The same as (7),  $s_j^{(0)}$  ( $j = n+1, \dots, n+m$ ) is normalized.

### Algorithm Introduction

In our algorithm, we label  $D^U$  by making use of information of both old domain and new domain. We fuse equations (3) and (4), and get the iterative equation as follows:

$$s_i^{(k)} = \alpha \sum_{j \in N_{i*}} (\hat{U}_{ij} \times s_j^{(k-1)}) + \beta \sum_{h \in M_{i*}} (\hat{V}_{ih} \times s_h^{(k-1)}), \quad i = 1, \dots, n \quad (8)$$

where  $\alpha + \beta = 1$ , and  $\alpha$  and  $\beta$  show the relative importance of old domain and new domain to the final sentiment scores. In consideration of the convergence,  $s_i^{(k)}$  is normalized after each iteration.

Here is the complete algorithm:

1. Classify  $D^U$  with a traditional classifier. Initialize the sentiment score  $s_i$  of  $d_i \in D^U \cup D^L$  ( $i = 1, \dots, n+m$ ) with 1 when  $d_i$  is labeled “positive”, and with -1 when  $d_i$  is labeled “negative”. And we normalize the sentiment scores to make the sum of positive scores of  $D^U$  ( $D^L$ ) equal to 1, and the sum of negative scores of  $D^U$  ( $D^L$ ) equal to -1.
2. Iteratively calculate the  $s_i^{(k)}$  of  $D^U$  and normalize it until it achieves the convergence:

$$s_i^{(k)} = \alpha \sum_{j \in N_{i*}} (\hat{U}_{ij} \times s_j^{(k-1)}) + \beta \sum_{h \in M_{i*}} (\hat{V}_{ih} \times s_h^{(k-1)}), \quad i = 1, \dots, n,$$

$$s_i^{(k)} = \begin{cases} s_i^{(k)} / \sum_{j \in D_{neg}^U} (-s_j^{(k)}), & \text{if } s_i^{(k)} < 0 \\ s_i^{(k)} / \sum_{j \in D_{pos}^U} s_j^{(k)}, & \text{if } s_i^{(k)} > 0 \end{cases} \quad i = 1, \dots, n$$

3. According to  $s_i \in S$  ( $i = 1, \dots, n$ ), assign each  $d_i \in D^U$  ( $i = 1, \dots, n$ ) a label. If  $s_i$  is between -1 and 0, assign  $d_i$  the label “negative”; if  $s_i$  is between 0 and 1, assign  $d_i$  the label “positive”.

## 4. EXPERIMENTS

### 4.1 Data Preparation

We prepare three Chinese domain-specific data sets from on-line reviews, which are: Electronics Reviews (Elec, from <http://detail.zol.com.cn/>), Stock Reviews (Stock, from <http://blog.sohu.com/stock/>) and Hotel Reviews (Hotel, from <http://www.ctrip.com/>). And then we manually label the reviews as “negative” or “positive”.

The detailed composition of the data sets are shown in Table 1, which shows the name of the data set (DataSet), the number of negative reviews (Neg), the number of positive reviews (Pos), the average length of reviews (Length), the number of different words (Vocabulary) in this data set.

Table 1. Data sets composition

DataSet	Neg	Pos	Length	Vocabulary
Elec	554	1,054	121	6,200
Stock	683	364	460	13,012
Hotel	2,000	2,000	181	11,336

Then, we use ICTCLAS (<http://ictclas.org/>), a Chinese text POS tool, to segment these Chinese reviews. The documents are represented by vector space model. We compute term weight with the frequency of the term in the document.

### 4.2 Evaluation Setup

In our experiment, we run prototype classification algorithm (Prototype, introduced in 3.4) and Support Vector Machine on the three data sets as our baselines. In our experiment, we use LibSVM ([www.csie.ntu.edu.tw/~cjlin/libsvm/](http://www.csie.ntu.edu.tw/~cjlin/libsvm/)) with a linear kernel and set all options by default.

We also compare our algorithm to Structural Correspondence Learning (SCL) [5]. SCL is a state-of-the-art sentiment-transfer algorithm which automatically induces correspondences among features from different domains. It identifies correspondences among features from different domains by modeling their correlations with pivot features, which are features that behave in the same way for discriminative learning in both domains. In our experiment, we use 100 pivot features.

### 4.3 Overall Performance

In this section, we conduct two groups of experiments where we separately initialize the sentiment scores in our algorithm by two different traditional classifiers: prototype classifier and Support Vector Machine.

There are two parameters in our algorithm,  $K$  and  $\alpha$  ( $\beta$  can be calculated by  $1 - \alpha$ ). We set the parame-

ters  $K$  and  $\alpha$  with 150 and 0.7 respectively, which indicates we use 150 neighbors and the contribution from old domain is a little more important than that from new domain. It is thought that the algorithm achieves the convergence when the changing between the sentiment score  $s_i$  computed at two successive iterations for any  $d_i \in D^U$  ( $i = 1, \dots, n$ ) falls below a given threshold, and we set the threshold 0.00001 in this work. These parameters will be studied in parameters sensitivity section.

Table 2 shows the accuracy of Prototype, LibSVM, SCL and our algorithm when training data and test data belong to different domains. Our algorithm is separately initialized by Prototype and LibSVM.

**Table 2. Accuracy comparison of different methods**

	Baseline		SCL	Proposed Algorithm	
	Proto-type	LibSVM		Prototype+SentiRank	LibSVM+SentiRank
Elec->Stock	0.6652	0.6478	<b>0.7507</b>	0.7326	0.7304
Elec->Hotel	0.7304	0.7522	<b>0.7750</b>	0.7543	0.7543
Stock->Hotel	0.6848	0.6957	<b>0.7682</b>	0.7435	0.7457
Stock->Elec	0.7043	0.6696	0.8339	<b>0.8457</b>	0.8435
Hotel->Stock	0.6196	0.5978	0.6571	<b>0.7848</b>	<b>0.7848</b>
Hotel->Elec	0.6674	0.6413	0.7269	<b>0.8609</b>	<b>0.8609</b>
Average	0.6786	0.6674	0.7520	<b>0.7870</b>	0.7866

As we can observe from Table 2, our algorithm can dramatically increase the accuracy of sentiment-transfer. Seen from the 2<sup>nd</sup> column and the 5<sup>th</sup> column, every accuracy of *SentiRank* is increased comparing to Prototype. The greatest increase of accuracy is achieved by about 19.5% on the problem “Hotel->Elec”. The second and third greatest increases of accuracy are achieved by about 16.5% and 14% on the problem “Hotel->Stock” and “Stock->Elec” respectively. The average increase of accuracy over all the 6 problems is 10.8%. Similarly, the accuracy of our algorithm is higher than LibSVM in every problem and the average increase of accuracy is 11.9%. The great improvement comparing with the baselines indicates that *SentiRank* algorithm performs very effectively and robustly.

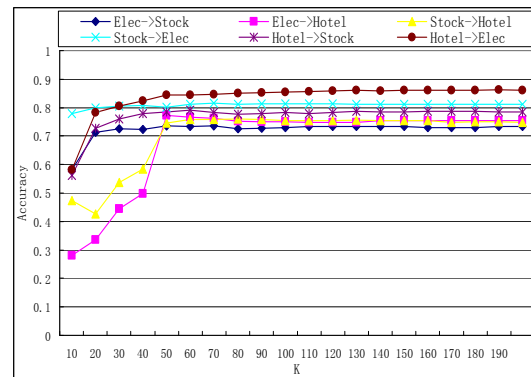
Seen from Table 2, our result about SCL is in accord with that in [5] on the whole. The average accuracy of SCL is higher than both baselines, which convinces that SCL is effective for sentiment-transfer. However, *SentiRank* outperforms SCL: the average accuracy of *SentiRank* is about 3.5% higher than SCL, and the greatest increase of accuracy even arrives at about 13.4% on the problem “Hotel->Elec”. This is caused by two reasons. First, SCL is essentially based on co-occurrence of words (the window size is the whole document), so it is easily affected by low frequency words and the size of data set. Second, the pivot features of SCL are totally dependent on experts in the field, so the quality of pivot

features will seriously affect the performance of SCL. This improvement convinces us of the effectiveness of our algorithm.

#### 4.4 Parameters Sensitivity

The proposed algorithm has two parameters,  $K$  and  $\alpha$  ( $\beta$  can be calculated by  $1 - \alpha$ ).  $K$  denotes the number of neighbors of a document we find to calculate its sentiment score.  $\alpha$  denotes the contribution of old domain to the sentiment score, and  $1 - \alpha$  denotes the contribution of new domain to the sentiment score. In this section, we conduct experiments to show that our algorithm is not sensitive to these parameters.

When we evaluate the parameter  $K$ , we set  $\alpha$  to 0.7 which indicates the contribution from old domain is a little more important than that from new domain. And we change  $K$  from 10 to 200, an increase of 10 each. We experiment the sensitivity of  $K$  with *SentiRank* initialized by Prototype on six problems as mentioned in section 4.3. And the results are shown in figure 2. It shows that our algorithm is not very sensitive to  $K$  as long as  $K$  is large enough. We find the curves of these six examples rise sharply when  $K$  increases from 10 to 50, and the curves become stable after  $K$  arrives at 100. So we set  $K$  as 150 in our overall-performance experiment.



**Figure 2. Accuracy for Different K**

To investigate the sensitivity of proposed method involved with the parameter  $\alpha$ , we set  $K$  to 150 which indicates we pick up 150 neighbors separately from both training domain and test domain to calculate the sentiment score. And we change  $\alpha$  from 0 to 1, an increase of 0.1 each. We also evaluate  $\alpha$  with *SentiRank* initialized by Prototype on the six problems mentioned above, and the results are shown in figure 3.

We can observe from Figure 3 that the accuracy first increases and then decreases when  $\alpha$  is increased from 0 to 1. The accuracy changes sharply when  $\alpha$  is near 0 or 1, and it changes less when  $\alpha$  is between 0.2 and 0.7. It is easy to explain this phenomenon. When  $\alpha$  is

set to 0, this indicates our algorithm only uses new domain (unlabelled data) to aid classification, without the accurate-labels information of old domain. And if  $\alpha$  is set to 1, our algorithm only uses old domain to calculate sentiment score, without the “pseudo” labels of new domain. Both cases above don’t use all information of both domains, so their accuracies are worse than to equal the contributions of both domains. This experiment shows that the proposed algorithm is not sensitive to the parameter  $\alpha$  as long as  $\alpha$  is not 0 or 1. We set  $\alpha$  as 0.7 in our overall-performance experiment.

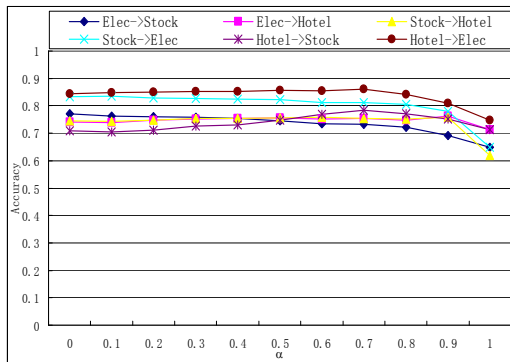


Figure 3. Accuracy for Different  $\alpha$

#### 4.5 Convergence

Our algorithm is an iterative process that will converge to a local optimum. We evaluate its convergence on the six problems mentioned above. Figure 4 shows the change of accuracy with respect to the number of iterations. We can observe from figure 4 that the curve rises sharply during the first 10 iterations, and it is nearly stable after 15 iterations are performed. This experiment indicates that our algorithm will be converged very quickly to get a local optimum.

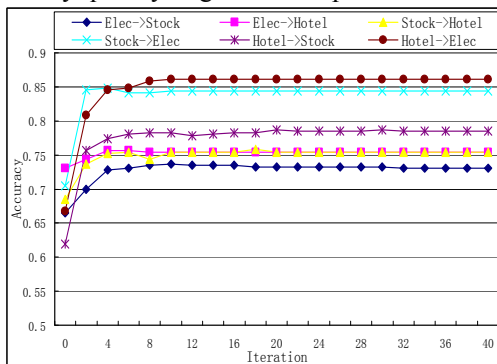


Figure 4. Performance for Iteration

## 5. Conclusion and Future Work

In this paper, we propose a novel cross-domain sentiment classification algorithm, namely *SentiRank*. It integrates the sentiment orientations of the documents into the graph-ranking based method for cross-domain sentiment classification.

The experiment results show that *SentiRank* can dramatically improve the accuracy when transferred to a new domain. Further more, it is observed that *SentiRank* is not very sensitive to its two parameters, and can be converged very quickly to get a local optimum.

## 6. Acknowledgments

This work was mainly supported by two funds, i.e., 0704021000 and 60803085, and two another projects, i.e., 2007CB311100 and 2007AA01Z441.

## 7. References

- [1] P. Turney. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In Proceedings of ACL 2002.
- [2] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In Proceedings of EMNLP, 2002.
- [3] S. Tan, X. Cheng, M. Ghanem, B. Wang and H. Xu. 2005. A Novel Refinement Approach for Text Categorization. In Proceedings of CIKM 2005.
- [4] A. Aue and M. Gamon. Customizing sentiment classifiers to new domains: a case study. In Proceedings of RANLP 2005.
- [5] J. Blitzer, M. Dredze, and F. Pereira. Biographies, Bollywood, Boom-boxes and Blenders: Domain adaptation for sentiment classification. In Proceedings of ACL 2007.
- [6] S. Tan, G. Wu, H. Tang and X. Cheng. 2007. A novel scheme for domain-transfer problem in the context of sentiment analysis. In Proceedings of CIKM 2007.
- [7] S. Tan, X. Cheng, Y. Wang, H. Xu. 2009. Adapting Naive Bayes to Domain Adaptation for Sentiment Analysis. In Proceedings of ECIR 2009.
- [8] H. Cui, V. Mittal, and M. Datar. 2006. Comparative experiments on sentiment classification for online product reviews. In Proceedings of AAAI-2006.
- [9] H. DaumeIII and D. Marcu. Domain adaptation for statistical classifiers. Journal of Artificial Intelligence Research, 2006, 26: 101-126.
- [10] J. Jiang, C. Zhai. A Two-Stage Approach to domain adaptation for statistical classifiers. In Proceedings of CIKM 2007.
- [11] S. Brin, L. Page, R. Motwami, and T. Winograd, The PageRank citation ranking: bringing order to the web, Technical Report 1999-0120. Stanford University. 1999.