MIEA: a Mutual Iterative Enhancement Approach for Cross-Domain Sentiment Classification

Qiong Wu[1,2], **Songbo Tan**[1], **Xueqi Cheng**[1] **and Miyi Duan**[1]
[1]Institute of Computing Technology, Chinese Academy of Sciences
[2] Graduate University of Chinese Academy of Sciences
{wuqiong,tansongbo}@software.ict.ac.cn

Abstract

Recent years have witnessed a large body of research works on cross-domain sentiment classification problem, where most of the research endeavors were based on a supervised learning strategy which builds models from only the labeled documents or only the labeled sentiment words. Unfortunately, such kind of supervised learning method usually fails to uncover the full knowledge between documents and sentiment words. Taking account of this limitation, in this paper, we propose an iterative reinforcement learning approach for cross-domain sentiment classification by simultaneously utilizing documents and words from both source domain and target domain. Our new method can make full use of the reinforcement between documents and words by fusing four kinds of relationships between documents and words. Experimental results indicate that our new method can improve the performance of cross-domain sentiment classification dramatically.

1 Introduction

Sentiment classification is the task of determining the opinion (e.g., negative or positive) of a given document. In recent years, it has drawn much attention with the increasing reviewing pages and blogs etc., and it is very important for many applications, such as opinion mining and summarization (e.g., (Ku et al., 2006; McDonald et al., 2007)).

In most cases, a variety of supervised classification methods can perform well in sentiment classification. This kind of methods requires a condition to guarantee the accuracy of classification: training data should have the same distribution with test data so that test data could share the information got from training data. So the labeled data in the same domain with test data is considered as the most valuable resources for the sentiment classification. However, such resources in different domains are very imbalanced. In some traditional domains or domains of concern, many labeled sentiment data are freely available on the web, but in other domains, labeled sentiment data are scarce and it involves much human labor to manually label reliable sentiment data. The challenge is how to utilize labeled sentiment data in one domain (that is, source domain) for sentiment classification in another domain (that is, target domain). This raises an interesting task, cross-domain sentiment classification (or sentiment transfer). In this work, we focus on one typical kind of sentiment transfer problem, which utilizes only training data from source domain to improve sentiment classification performance for target domain, without any labeled data for the target domain (e.g., (Andreevskaia and Bergler, 2008)).

In recent years, some studies have been conducted to deal with sentiment transfer problems. However, most of the attempts rely on only the labeled documents (Aue and Gamon, 2005; Tan et al., 2007; Tan et al., 2009; Wu et al., 2009) or the labeled sentiment words (Gamon and Aue, 2005) to improve the performance of sentiment transfer, so this kind of methods fails to uncover the full knowledge between the documents and the sentiment words.

In fact, the opinion of a document can be determined by the interrelated documents as well as by the interrelated words, and this rule is also tenable when determining the opinion of a sentiment word. This rule is based on the following intuitive observations:

(1) A document strongly linked with other positive (negative) documents could be considered as positive (negative); in the same way, a word strongly linked with other positive (negative) words could be considered as positive (negative).

(2) A document containing many positive (negative) words could be considered as positive (negative); similarly, a word appearing in many positive (negative) documents could be considered as positive (negative).

Inspired by these observations, we aim to take into account all the four kinds of relationships among documents and words (i.e. the relationships between documents, the relationships between words, the relationships between words and documents, and the relationships between documents and words) in both source domain and target domain under a unified framework for sentiment transfer.

In this work, we propose an iterative reinforcement approach to implement the above idea. The proposed approach makes full use of all the relationships among documents and words from both source domain and target domain to transfer information between domains. In our approach, the opinion of a document (word) is reinforced by the opinion of all its interrelated documents and words; and the updated opinion of the document (word) will conversely reinforce the opinions of its interrelated documents and words. That is to say, it is an iterative reinforcement process until it converges to a final result.

The contribution of our work is twofold. First, we extend the traditional sentiment-transfer methods by utilizing the full knowledge between interrelated documents and words. Second, we present a reinforcement approach to get the opinions of documents by making use of graph-ranking algorithm.

The proposed approach is evaluated on three domain-specific sentiment data sets. The experimental results show that our approach can dramatically improve the accuracy when transferred to another target domain. And we also conduct extensive experiments to investigate the parameters sensitivity. The results show that our algorithm is not sensitive to these parameters.

2 Proposed Methods

2.1 Problem Definition

In this paper, we have two document sets: the test documents $D^U = \{d_1,\ldots,d_{nd}\}$ where d_i is the term vector of the i^{th} text document and each $d_i \in D^U (i = 1,\ldots,nd)$ is unlabeled; the training documents $D^L = \{d_{nd+1},\ldots,d_{nd+md}\}$ where d_j represents the term vector of the j^{th} text document and each $d_j \in D^L (j = nd+1,\ldots,nd+md)$ should have a label from a category set C = {negative, positive}. We assume the training dataset D^L is from the interrelated but different domain with the test dataset D^U. Also, we have two word sets: $W^U = \{w_1,\ldots,w_{nw}\}$ is the word set of D^U and each $w_i \in W^U$ $(i = 1,\ldots,nw)$ is unlabeled; $W^L = \{w_{nw+1},\ldots,w_{nw+mw}\}$ is the word set of D^L and each $w_j \in W^L (j = nw+1,\ldots,nw+mw)$ has a label from C. Our objective is to maximize the accuracy of assigning a label in C to $d_i \in D^U$ $(i = 1,\ldots,nd)$ utilizing the training data D^L and W^L in another domain.

The proposed algorithm is based on the following presumptions:

(1) $W^L \cap W^U \neq \Phi$.

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

2.2 Overview

The proposed approach is inspired by graph-ranking algorithm whose idea is to give a node high score if it is strongly linked with other high-score nodes. Graph-ranking algorithm has been successfully used in many fields (e.g. PageRank (Brin et al, 1999), LexRank (Erkan and Radev, 2004)). We can get the following thoughts based on the ideas of PageRank and HITS (Kleinberg, 1998):

(1) If a document is strongly linked with other positive (negative) documents, it tends to be positive (negative); and if a word is strongly linked with other positive (negative) words, it tends to be positive (negative).

(2) If a document contains many positive (negative) words, it tends to be positive (negative); and if a word appears in many positive (negative) documents, it tends to be positive (negative).

Given the data points of documents and words, there are four kinds of relationships in our problem:

- DD-Relationship: It denotes the relationships between documents, usually computed by their content similarity.
- WW-Relationship: It denotes the relationships between words, usually computed by knowledge-based approach or corpus-based approach.
- DW-Relationship: It denotes the relationships between documents and words, usu-

ally computed by the relative importance of a word in a document.

- WD-Relationship: It denotes the relationships between words and documents, usually computed by the relative importance of a document to a word.

Meanwhile, our problem refers to both source domain and target domain, so our approach considers eight relationships altogether: DDO-Relationship (the relationships between D^U and D^L), DDN-Relationship (the relationships between D^U), WWO-Relationship (the relationships between W^U and W^L), WWN-Relationship (the relationships between W^U and W^U), DWO-Relationship (the relationships between D^U and W^L), DWN-Relationship (the relationships between D^U and W^U), WDO-Relationship (the relationships between W^U and D^L), WDN-Relationship (the relationships between W^U and D^U). The first four relationships are used to compute the sentiment scores of the documents, and the others are used to compute the sentiment scores of the words.

The iterative reinforcement approach could make full use of all the relationships in a unified framework. The framework of the proposed approach is illustrated in Figure 1.

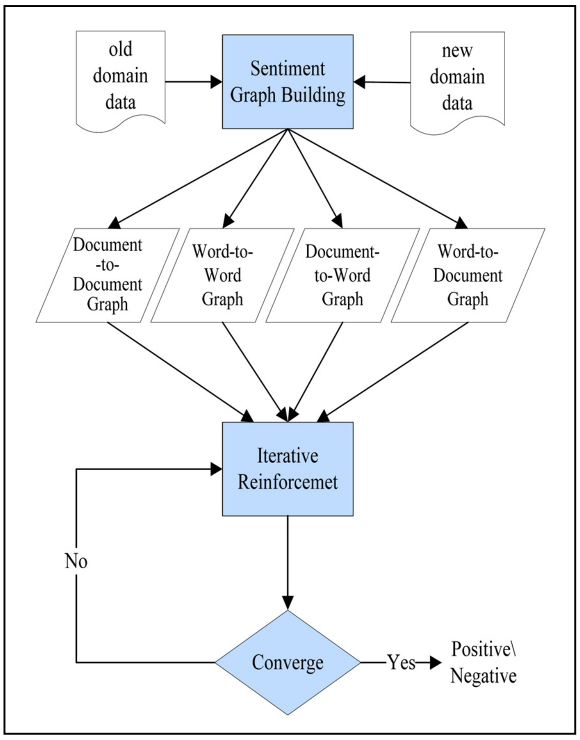

Figure 1. Framework of the proposed approach

The framework consists of a graph-building phase and an iterative reinforcement phase. In the graph-building phase, the input includes both the labeled data from source domain and the unlabeled data from target domain. The proposed approach builds four graphs based on these data to reflect the above relationships respectively. For source-domain data, we initialize every document and word a score (“1” denotes positive, and “-1” denotes negative) to represent its degree of sentiment orientation, and we call it sentiment score; for target-domain data, we set the initial sentiment scores to 0.

In the iterative reinforcement phase, our approach iteratively computes the sentiment scores of the documents and words based on the graphs. When the algorithm converges, all the documents get their sentiment scores. If 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, if 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.

The algorithms of sentiment graph building and iterative reinforcement are described in details in the next sections, respectively.

2.3 Sentiment-Graph Building

Symbol Definition

In this section, we build four graphs to reflect eight relationships, and the meanings of symbols are shown in Table 1.

Relationship	Similarity matrix	Normalized form	Neighbor matrix
DDO	$U^L=[U^L{}_{ij}]_{ndxmd}$	$\hat{U}^L$	$Un^L=[Un^L{}_{ij}]_{nd\times K}$
DDN	$U^U=[U^U{}_{ij}]_{ndxnd}$	$\hat{U}^U$	$Un^U=[Un^U{}_{ij}]_{nd\times K}$
WWO	$V^L=[V^L{}_{ij}]_{nwxmw}$	$\hat{V}^L$	$Vn^L=[Vn^L{}_{ij}]_{nw\times K}$
WWN	$V^U=[V^U{}_{ij}]_{nwxnw}$	$\hat{V}^U$	$Vn^U=[Vn^U{}_{ij}]_{nw\times K}$
DWO	$M^L=[M^L{}_{ij}]_{ndxmw}$	$\hat{M}^L$	$Mn^L=[Mn^L{}_{ij}]_{nd\times K}$
DWN	$M^U=[M^U{}_{ij}]_{ndxnw}$	$\hat{M}^U$	$Mn^U=[Mn^U{}_{ij}]_{nd\times K}$
WDO	$N^L=[N^L{}_{ij}]_{nwxmd}$	$\hat{N}^L$	$Nn^L=[Nn^L{}_{ij}]_{nw\times K}$
WDN	$N^U=[N^U{}_{ij}]_{nwxnd}$	$\hat{N}^U$	$Nn^U=[Nn^U{}_{ij}]_{nw\times K}$

Table 1: Symbol definition

In this table, the first column denotes the name of the relationship; the second column denotes the similarity matrix to reflect the corresponding relationship; in consideration of convergence, we normalize the similarity matrix, and the normalized form is listed in the third column; in order to compute sentiment scores, we find the neighbors of a document or a word and the neighbor matrix is listed in the fourth column.

Document-to-Document Graph

We build an undirected 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 edges in this graph are divided into two parts: edges between D^U and D^L; edges between D^U itself, so we build the graph in two steps.

(1) Create D^U and D^L Edges

The content similarity between two documents is computed with the cosine measure. We use an adjacency matrix U^L to denote the similarity matrix between D^U and D^L. $U^L=[U^L_{ij}]_{nd \times md}$ is defined as follows:

$$U^L_{ij} = \frac{d_i \bullet d_{j+nd}}{\|d_i\| \times \|d_{j+nd}\|},\ i = 1,...,nd, j = 1,...,md \qquad (1)$$

The weight associated with word w is computed with $tf_w idf_w$ where tf_w is the frequency of word w in the document and idf_w is the inverse document frequency of word w, i.e. $1+\log(N/n_w)$, where N is the total number of documents and n_w is the number of documents containing word w in a data set.

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

$$\hat{U}^L_{ij} = \begin{cases} U^L_{ij} \Big/ \sum_{j=1}^{md} U^L_{ij},\ if \sum_{j=1}^{md} U^L_{ij} \neq 0 \\ 0,\ otherwise \end{cases} \qquad (2)$$

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

Then for $d_i \in D^U$ ($i = 1,\dots,nd$), $\tilde{U}^L_{ij}$ ($j = 1,\dots,K$) corresponds to K neighbors in D^L. We use a matrix $Un^L = [Un^L_{ij}]_{nd \times K}$ to denote the neighbors of D^U in source domain, with Un^L_{ij} corresponding to the j^{th} nearest neighbor of d_i.

(2) Create D^U and D^U Edges

Similarly, the edge weight between D^U itself is computed by the cosine measure. We get the similarity matrix $U^U=[U^U_{ij}]_{nd \times nd}$, the normalized similarity matrix $\hat{U}^U$, and the neighbors of D^U in target domain: $Un^U = [Un^U_{ij}]_{nd \times K}$.

Word-to-Word Graph

Similar to the Document-to-Document Graph, we build an undirected graph to reflect the relationship between words in W^L and W^U, in which each node corresponds to a word and the edge weight between any different words corresponds to their semantic similarity. The edges in this graph are divided into two parts: edges between W^U and W^L; edges between W^U itself, so we also build the graph in two steps.

(1) Create W^U and W^L Edges

We compute the semantic similarity using corpus-based approach which computes the similarity between words utilizing information from large corpora. There are many measures to identify word semantic similarity, such as mutual information (Turney, 2001), latent semantic analysis (Landauer et al., 1998) etc. In this study, we compute word semantic similarity based on the sliding window measure, that is, two words are semantically similar if they co-occur at least once within a window of maximum K_{win} words, where K_{win} is the window size. We use an adjacency matrix V^L to denote the similarity matrix between W^U and W^L. $V^L=[V^L_{ij}]_{nw \times mw}$ is defined as follows:

$$V^L_{ij} = \begin{cases} \log \frac{N \times p(w_i, w_{j+nw})}{p(w_i) \times p(w_{j+nw})},\ if\ w_i \neq w_{j+nw} \\ 0,\ otherwise \end{cases} \qquad (3)$$

where N is the total number of words in D^U; $p(w_i, w_j)$ is the probability of the co-occurrence of w_i and w_j within a window, i.e. $num(w_i, w_j)/N$, where $num(w_i, w_j)$ is the number of the times w_i and w_j co-occur within the window; $p(w_i)$ and $p(w_j)$ are the probabilities of the occurrences of w_i and w_j respectively, i.e. $num(w_i)/N$ and $num(w_j)/N$, where $num(w_i)$ and $num(w_j)$ are the numbers of the times w_i and w_j occur. We normalize V^L to $\hat{V}^L$ to make the sum of each row equal to 1. Then we sort every row of $\hat{V}^L$ to $\tilde{V}^L$ in descending order, and we use a matrix $Vn^L = [Vn^L{}_{ij}]_{nw \times K}$ to denote the neighbors of W^U in source domain.

(2) Create W^U and W^U Edges

Then we also compute the edge weight between any different nodes which denote words in W^U by the sliding window measure. We get the similarity matrix $V^U=[V^U{}_{ij}]_{nwxnw}$, the normalized similarity matrix $\hat{V}^U$, and the neighbors of W^U in target domain: $Vn^U = [Vn^U{}_{ij}]_{nw \times K}$.

Document-to-Word Graph

We can build a weighted directed bipartite graph from documents in D^U and words in W^L and W^U in the following way: each node in the graph corresponds to a document in D^U or a word in W^L and W^U; if word w_j appears in document d_i, we create an edge from d_i to w_j. The edges in this graph are divided into two parts: edges from D^U to W^L; edges from D^U to W^U, so we also build the graph in two steps.

(1) Create D^U to W^L Edges

The edge weight from a document in D^U to a word in W^L is proportional to the importance of word w_j in document d_i. We use an adjacency matrix M^L to denote the similarity matrix from D^U to W^L. $M^L=[M^L{}_{ij}]_{ndxmw}$ is defined as follows:

$$M^L_{ij} = \frac{tf_{w_{j+nw}} \times idf_{w_{j+nw}}}{\sum_{w \in d_i} tf_w \times idf_w} \quad (4)$$

where w represents a unique word in d_i and tf_w, idf_w are respectively the term frequency in the document and the inverse document frequency. We normalize M^L to $\hat{M}^L$ to make the sum of each row equal to 1. Then we sort every row of $\hat{M}^L$ to $\tilde{M}^L$ in descending order, and we use a matrix $Mn^L = [Mn^L{}_{ij}]_{nd \times K}$ to denote the neighbors of D^U in W^L.

(2) Create D^U to W^U Edges

Similarly, we can also compute the edge weight from a document in D^U to a word in W^U in the same way. We get the similarity matrix $M^U=[M^U{}_{ij}]_{ndxnw}$, the normalized similarity matrix $\hat{M}^U$, and the neighbors of D^U in W^U: $Mn^U = [Mn^U{}_{ij}]_{nd \times K}$.

Word-to-Document Graph

In this section, we build a weighted directed bipartite graph from words in W^U and documents in D^L and D^U in which each node in the graph corresponds to a word in W^U and a document in D^L or D^U; if word w_j appears in document d_i, we create an edge from w_j to d_i. The edges in this graph are also divided into two parts: edges from W^U to D^L; edges from W^U to D^U.

(1) Create W^U to D^L Edges

Similar to 3.3.4, the edge weight from a word in W^U to a document in D^L is proportional to the importance of word w_i in document d_j. We use an adjacency matrix $N^L=[N^L{}_{ij}]_{nwxmd}$ to denote the similarity matrix from W^U to D^L. We normalize N^L to $\hat{N}^L$ to make the sum of each row equal to 1. Then we sort every row of $\hat{N}^L$ to $\tilde{N}^L$ in descending order, and we use a matrix $Nn^L = [Nn^L{}_{ij}]_{nw \times K}$ to denote the neighbors of W^U in D^L.

(2) Create W^U to D^U Edges

We can also compute the edge weight from a word in W^U to a document in D^U in the same way. We get the similarity matrix $N^U=[N^U{}_{ij}]_{nwxnd}$, the normalized similarity matrix $\hat{N}^U$, and the neighbors of W^U in D^U: $Nn^U = [Nn^U{}_{ij}]_{nw \times K}$.

2.4 Proposed Method

Based on the two thoughts introduced in Section 2.2, we fuse the eight relationships abstracted from the four graphs together to iteratively reinforce sentiment scores, and we can obtain the iterative equation as follows:

$$\begin{aligned} ds_i = \varphi \sum_{g \in Un^L{}_{i\bullet}} (\hat{U}^L{}_{ig} \times ds_g) + \mu \sum_{h \in Un^U{}_{i\bullet}} (\hat{U}^U{}_{ih} \times ds_h) \\ + \gamma \sum_{l \in Mn^L{}_{i\bullet}} (\hat{M}^L{}_{il} \times ws_l) + \delta \sum_{r \in Mn^U{}_{i\bullet}} (\hat{M}^U{}_{ir} \times ws_r) \end{aligned} \quad (5)$$

$$\begin{aligned} ws_j = \varphi \sum_{g \in Vn^L{}_{j\bullet}} (\hat{V}^L{}_{jg} \times ws_g) + \mu \sum_{h \in Vn^U{}_{j\bullet}} (\hat{V}^U{}_{jh} \times ws_h) \\ + \gamma \sum_{l \in Nn^L{}_{j\bullet}} (\hat{N}^L{}_{jl} \times ds_l) + \delta \sum_{r \in Nn^U{}_{j\bullet}} (\hat{N}^U{}_{jr} \times ds_r) \end{aligned} \quad (6)$$

where $i\bullet$ means the i^{th} row of a matrix; $Ds = \{ds_1,\ldots,ds_{nd}, ds_{nd+1},\ldots, ds_{nd+md}\}$ represents the sentiment scores of D^U and D^L; $Ws = \{ws_1,\ldots,ws_{nw}, ws_{nw+1},\ldots, ws_{nw+mw}\}$ represents the sentiment scores of W^U and W^L; φ and μ show the relative contributions to the final sentiment scores from source domain and target domain when calculating DD-Relationship and WW-Relationship, and $\varphi + \mu = 1$; γ and δ show the relative contributions to the final sentiment scores from source domain and target domain when calculating DW-Relationship and WD-Relationship, and $\gamma + \delta = 1$.

For simplicity, we merge the relationships from source domain and target domain. That is, for formula (5), we merge the first two items into one, the last two items into one; for formula (6), we merge its first two items into one, its last two items into one. Thus, (5) and (6) are transformed into (7) and (8) as follows:

$$ds_i = \alpha \times \sum_{g \in Un_{i\bullet}} (\hat{U}_{ig} \times ds_g) + \beta \times \sum_{l \in Mn_{i\bullet}} (\hat{M}_{il} \times ws_l) \qquad (7)$$

$$ws_j = \alpha \times \sum_{g \in Nn_{j\bullet}} (\hat{N}_{jg} \times ds_g) + \beta \times \sum_{l \in Vn_{j\bullet}} (\hat{V}_{jl} \times ws_l) \qquad (8)$$

where α and β show the relative contributions to the final sentiment scores from document sets and word sets, and $\alpha + \beta = 1$.

In consideration of the convergence, *Ds* and *Ws* are normalized separately after each iteration as follows to make the sum of positive scores equal to 1, and the sum of negative scores equal to -1:

$$ds_i = \begin{cases} ds_i \Big/ \sum_{j \in D^U_{neg}} (-ds_j), & if \;\; ds_i < 0 \\ ds_i \Big/ \sum_{j \in D^U_{pos}} ds_j, & if \;\; ds_i > 0 \end{cases} \qquad (9)$$

$$ws_j = \begin{cases} ws_j \Big/ \sum_{i \in W^U_{neg}} (-ws_i), & if \;\; ws_j < 0 \\ ws_j \Big/ \sum_{i \in W^U_{pos}} ws_i, & if \;\; ws_j > 0 \end{cases} \qquad (10)$$

where D^U_{neg} and D^U_{pos} denote the negative and positive document set of D^U respectively; W^U_{neg} and W^U_{pos} denote the negative and positive word set of W^U respectively.

Here is the complete algorithm:

1. Initialize the sentiment score *vector* ds_i of $d_i \in D^L$ ($i = nd+1,\ldots, nd+md$) with 1 when d_i is labeled "positive", and with -1 when d_i is labeled "negative", and initialize the sentiment score *vector* ws_i of $w_i \in W^L$ (i = nw+1,…, nw+mw) with 1 when w_i is labeled "positive", and with -1 when w_i is labeled "negative". And we normalize ds_i (i = nd+1,…, nd+md) (ws_i (i = nw+1,…, nw+mw)) to make the sum of positive scores of D^L (W^L) equal to 1, and the sum of negative scores of D^L (W^L) equal to -1. Also, the initial sentiment scores of D^U and W^U are set to 0.
2. Alternate the following two steps until convergence:
   2.1.Compute and normalize ds_i ($i = 1,\ldots, nd$) using formula (7) and (9):
   2.2.Compute and normalize ws_j ($j=1,\ldots,nw$) using formula (8) and (10):

   where $ds_i^{(k)}$ and $ws_j^{(k)}$ denote the ds_i and ws_j at the k^{th} iteration.
3. According to $ds_i \in Ds$ ($i = 1,\ldots,nd$), assign each $d_i \in D^U$ ($i = 1,\ldots,nd$) a label. If ds_i falls in the range [-1,0], assign d_i the label "negative"; if ds_i falls in the range [0,1], assign d_i the label "positive".

3 Experiments

In this section, we evaluate our approach on three different domains and compare it with some state-of-the-art algorithms, and also evaluate the approach's sensitivity to its parameters. Note that we conduct experiments on Chinese data, but the main idea in the proposed approach is language-independent in essence.

3.1 Data Preparation

We use three Chinese domain-specific data sets from on-line reviews, which are: Book Reviews[1] (B, www.dangdang.com/), Hotel Reviews[2] (H, www.ctrip.com/) and Notebook Reviews[3] (N, www.360buy.com/). Each dataset has 4000 labeled reviews (2000 positives and 2000 negatives).

We use ICTCLAS (http://ictclas.org/), a Chinese text POS tool, to segment these Chinese reviews. Then, utilizing the part-of-speech tagging function provided by ICTCLAS, we take all adjectives, adverbs and adjective-noun phrases as candidate sentiment words. After removing the repeated words and ambiguous words, we get a list of words in each domain.

For the list of words in each domain, we manually label every word as "negative", "posi-

[1]www.searchforum.org.cn/tansongbo/corpus/Dangdang_Book_4000.rar
[2]www.searchforum.org.cn/tansongbo/corpus/Ctrip_htl_4000.rar
[3]www.searchforum.org.cn/tansongbo/corpus/Jingdong_NB_4000.rar

tive" or "neutral", and we take those "negative" and "positive" words as a sentiment word set.

Note that we use the sentiment word set only for source domain, while using the candidate sentiment words for target domain.

Lastly, the documents are represented by vector space model. In this model, each document is converted into bag-of-words presentation in the remaining term space. We compute term weight with the frequency of the term in the document.

We choose one of the three data sets as source-domain data D^L, and its corresponding sentiment word set as W^L; we choose another data set as target-domain data D^U, and its corresponding candidate sentiment words as W^U.

3.2 Baseline Methods

In this paper we compare our approach with the following baseline methods:

Proto: This method applies a traditional supervised classifier, prototype classifier (Tan et al., 2005), for the sentiment transfer. And it only uses source domain documents as training data.

LibSVM: This method applies a state-of-the-art supervised learning algorithm, Support Vector Machine, for the sentiment transfer. In detail, we use LibSVM (Chang and Lin, 2001) with a linear kernel and set all options as default. This method only uses source domain documents as training data.

TSVM: This method applies transductive SVM (Joachims, 1999) for the sentiment transfer which is a widely used method for improving the classification accuracy. In our experiment, we use Joachims's SVM-light package (http://svmlight.joachims.org/) for TSVM. We use a linear kernel and set all parameters as default. This method uses both source domain data and target domain data.

3.3 Overall Performance

In this section, we compare proposed approach with the three baseline methods. There are three parameters in our algorithm, K, K_{win}, α (β can be calculated by 1-α). We set K to 50, and K_{win} to 10 respectively. With different α, our approach can be considered as utilizing different relative contributions from document sets and word sets. In order to identify the importance of both document sets and word sets for sentiment transfer, we separately set α to 0, 1, 0.5 to show the accuracy of utilizing only word sets (referred to as WORD), only document sets (referred to as DOC), and both the document and word sets (referred to as ALL). It is thought that the algorithm achieves the convergence when the changing between the sentiment score ds_i computed at two successive iterations for any $d_i \in D^U$ (i = 1,…,nd) falls below a given threshold, and we set the threshold 0.00001 in this work. The parameters will be studied in parameters sensitivity section.

Table 2 shows the accuracy of Prototype, LibSVM, TSVM and our algorithm when training data and test data belong to different domains.

As we can observe from Table 2, our algorithm produces much better performance than supervised baseline methods. Compared with the traditional classifiers, our approach outperforms them by a wide margin on all the six transfer tasks. The great improvement compared with the baselines indicates that our approach performs very effectively and robustly.

	Traditional Classifier		TSVM	Our Approach		
	Proto	LibSVM		DOC	WORD	ALL
B->H	0.735	0.747	0.749	**0.772**	0.734	0.763
B->N	0.651	0.652	0.769	0.714	0.785	**0.795**
H->B	0.645	0.675	0.614	0.671	0.668	**0.703**
H->N	0.729	0.669	0.726	**0.749**	0.727	0.734
N->B	0.612	0.608	0.622	0.638	0.667	**0.726**
N->H	0.724	0.711	0.772	0.764	0.740	**0.792**
Average	0.683	0.677	0.709	0.718	0.720	**0.752**

Table 2: Accuracy comparison of different methods

Table 2 shows the average accuracy of TSVM is higher than both traditional classifiers, since it utilizes the information of both source domain and target domain. However, the proposed approach outperforms TSVM: the average accuracy of the proposed approach is about 4.3% higher than TSVM. This is caused by two reasons. First, TSVM is not dedicated for sentiment-transfer learning. Second, TSVM requires the ratio between positive and negative examples in the test data to be close to the ratio in the training data, so its performance will be affected if this requirement is not met.

Results of "DOC" and "WORD" are shown in column 4 and 5 of Table 2. As we can observe, they produce better performance than all the baselines. This is caused by two reasons. First, "DOC" and "WORD" separately utilize the sentiment information of documents and words. Second, both "DOC" and "WORD" involve an iterative reinforcement process to improve their performance. The great improvement indicates that the iterative reinforcement approach is effective for sentiment transfer.

Besides, Table 2 also shows both document sets and word sets are important for sentiment transfer. The approach "ALL" outperforms the approaches "DOC" and "WORD" on almost all the six transfer tasks except "B->H" and "H->N". The average increase of accuracy over all the six tasks is 3.4% and 3.2% respectively. The reason is: at every iteration, the classification accuracy of documents and words is improved by each other, and then the accuracy of sentiment transfer is improved by the documents and words that are classified more accurately. As for "B->H" and "H->N", the performance of utilizing only document sets is so good that the word sets couldn't improve the performance any more. The improvement of the approach "ALL" convinces us that not a single one of the four relationships can be omitted.

3.4 Parameters Sensitivity

The proposed algorithm has an important parameter, α (β can be calculated by 1-α). In this section, we conduct experiments to show that our algorithm is not sensitive to this parameter.

To investigate the sensitivity of proposed method involved with the parameter α, we set K to 50, and K_{win} to 10. And we change α from 0 to 1, an increase of 0.1 each. We also evaluate α on the six tasks mentioned in section 3.1, and the results are shown in figure 2.

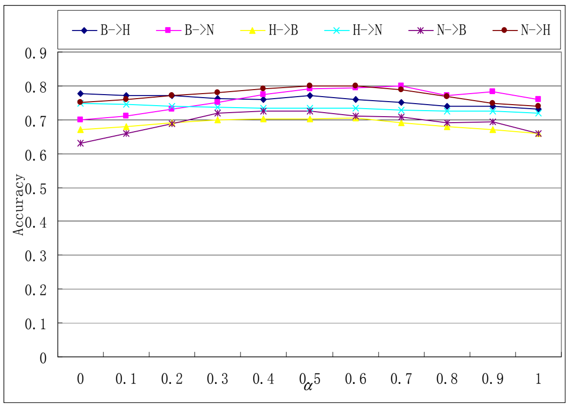

Figure 2: Accuracy for Different α

We can observe from Figure 2 that the accuracy first increases and then decreases when α is increased from 0 to 1. The accuracy changes gradually when α is near 0 or 1, and it changes less when α is between 0.2 and 0.8. It is easy to explain this phenomenon. When α is set to 0, this indicates our algorithm only uses word sets to aid classification, without the information of document sets. And if α is set to 1, our algorithm only uses document sets to calculate sentiment score, without the help of word sets. Both cases above don't use all information of four relationships, so their accuracies are worse than to equal the contributions of both document and word sets. This experiment shows that the proposed algorithm is not sensitive to the parameter α as long as α is not 0 or 1. We set α to 0.5 in our overall-performance experiment.

3.5 Convergence

Our algorithm is an iterative process that will converge to a local optimum. We evaluate its convergence on the six tasks mentioned above. Figure 3 shows the change of accuracy with respect to the number of iterations. We can observe from figure 3 that the curve rises sharply during the first 6 iterations, and it is very stable after 10 iterations are performed. This experiment indicates that our algorithm could converge very quickly to get a local optimum.

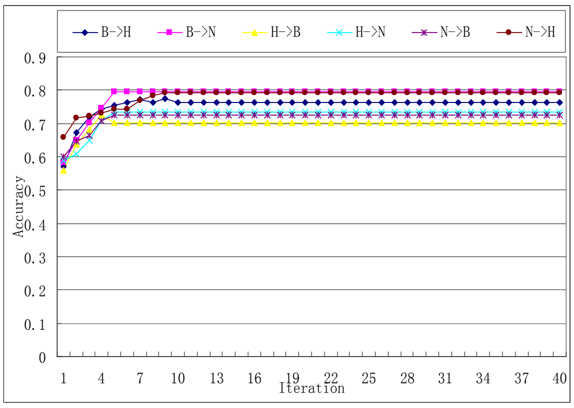

Figure 3: Performance for Iteration

4 Conclusions

In this paper, we propose a novel cross-domain sentiment classification approach, which is an iterative reinforcement approach for sentiment transfer by utilizing all the relationships among documents and words from both source domain and target domain to transfer information between domains. First, we build three graphs to reflect the above relationships respectively. Then, we assign a score for every unlabelled document to denote its extent to “negative” or “positive”. We then iteratively calculate the score by making use of the graphs. Finally, the final score for sentiment classification is achieved when the algorithm converges, so we can label the target-domain data based on these scores.

We conduct experiments on three domain-specific sentiment data sets. The experimental results show that the proposed approach could dramatically improve the accuracy when transferred to a target domain. To investigate the parameter sensitivity, we conduct experiments on the same data sets. It is observed that our approach is not very sensitive to its four parameters, and could converge very quickly to get a local optimum.

In this study, we employ only cosine measure, sliding window measure and vector measure to compute similarity. These are too general, and perhaps not so suitable for sentiment classification. In the future, we will try other methods to calculate the similarity. Furthermore, we experiment our approach on only three domains, and we will apply our approach to many more domains.

5 Acknowledgments

This work was mainly supported by two funds, i.e., 60933005 & 60803085, and two another projects, i.e., 2007CB311100 & 2007AA01Z441.

References

Alina Andreevskaia and Sabine Bergler. 2008. When Specialists and Generalists Work Together: Overcoming Domain Dependence in Sentiment Tagging. In Proceedings of ACL: 290-298.

Anthony Aue and Michael Gamon. 2005. Customizing sentiment classifiers to new domains: a case study. In Proceedings of RANLP.

Sergey Brin, Lawrence Page, Rajeev Motwami, and Terry Winograd. 1999. The PageRank citation ranking: bringing order to the web. Technical Report 1999-0120, Stanford, CA.

Chinchung Chang and Chinjen Lin. 2001. LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Gunes Erkan and Dragomir Radev. 2004. LexRank: Graph-based Centrality as Salience in Text Summarization. Journal of Artificial Intelligence Research, 22 (2004): 457-479.

Michael Gamon and Anthony Aue. 2005. Automatic identification of sentiment vocabulary: exploiting low association with known sentiment terms. In Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in NLP: 57-64.

Songbo Tan, Xueqi Cheng, Moustafa Ghanem, Bin Wang, Hongbo Xu. 2005. A novel refinement approach for text categorization. In Proceedings of CIKM 2005: 469-476

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In Proceedings of ICML.

Jon Kleinberg. 1998. Authoritative sources in a hyperlinked environment. Journal of the ACM, 46(5): 604-632.

Lunwei Ku, Yuting Liang, and Hsinhsi Chen. 2006. Opinion extraction, summarization and tracking in news and blog corpora. In Proceedings of AAAI.

Thomas Landauer, Peter Foltz, and Darrell Laham. 1998. Introduction to latent semantic analysis. Discourse Processes 25: 259-284.

Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In Proceedings of ACL.

Songbo Tan, Yuefen Wang, Gaowei Wu, and Xueqi Cheng. 2007. A novel scheme for domain-transfer problem in the context of sentiment analysis. In Proceedings of CIKM.

Songbo Tan, Xueqi Cheng, Yuefen Wang, and Hongbo Xu. 2009. Adapting Naïve Bayes to Domain Adaptation for Sentiment Analysis. In Proceedings of ECIR.

Qiong Wu, Songbo Tan and Xueqi Cheng. 2009. Graph Ranking for Sentiment Transfer. In Proceedings of ACL-IJCNLP.

Peter Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In Proceedings of ECML: 491-502.