

Microblog Sentiment Classification with Contextual Knowledge Regularization

Fangzhao Wu[†], Yangqiu Song[‡], Yongfeng Huang[†]

[†]Tsinghua National Laboratory for Information Science and Technology,
Department of Electronic Engineering, Tsinghua University, Beijing, China

[‡]University of Illinois at Urbana-Champaign, Urbana, IL, USA
wufangzhao@gmail.com, yqsong@illinois.edu, yfhuang@tsinghua.edu.cn

Abstract

Microblog sentiment classification is an important research topic which has wide applications in both academia and industry. Because microblog messages are short, noisy and contain masses of acronyms and informal words, microblog sentiment classification is a very challenging task. Fortunately, collectively the contextual information about these idiosyncratic words provide knowledge about their sentiment orientations. In this paper, we propose to use the microblogs' contextual knowledge mined from a large amount of unlabeled data to help improve microblog sentiment classification. We define two kinds of contextual knowledge: word-word association and word-sentiment association. The contextual knowledge is formulated as regularization terms in supervised learning algorithms. An efficient optimization procedure is proposed to learn the model. Experimental results on benchmark datasets show that our method can consistently and significantly outperform the state-of-the-art methods.

Introduction

Microblogging services, such as Twitter, have become very popular in recent years. They provide public platforms for users to share their opinions on various topics, such as daily living activities, social or political events, news about companies or celebrities and so on. Identifying sentiments or opinions from microblogs can statistically facilitate or validate many other disciplines, including social customer relationship management, political science, and social psychology etc. (Go, Bhayani, and Huang 2009; O'Connor et al. 2010; Bollen, Mao, and Pepe 2011; Wu et al. 2014).

Machine learning methods, especially supervised learning methods, are widely used in microblog sentiment classification field (Go, Bhayani, and Huang 2009; Birmingham and Smeaton 2010; Liu, Li, and Guo 2012; Hu et al. 2013). These methods use labeled data to train a sentiment classifier to classify the new microblog messages. However, microblog messages are usually very short and noisy, and contain massive acronyms and informal words, such as "tnx" and "coooooo!" This brings challenges to microblog sentiment classification, because the labeled training data may

not be sufficient to infer a model to predict the sentiment polarity for such acronyms and informal words. Manually labeling enough data is costly and time consuming. On the other hand, the unlabeled data are relatively cheap and microblogs in their nature provide a lot of knowledge about sentiment orientations of the short messages. For example, microblogs usually enable and encourage users to use emoticons to express their emotions. Thus, it will be helpful to mine such sentiment knowledge from unlabeled data to improve classification.

One way to use the knowledge from large scale unlabeled data is to build a larger sentiment lexicon to increase the coverage (Wilson, Wiebe, and Hoffmann 2005; Baccianella, Esuli, and Sebastiani 2010; Dang, Zhang, and Chen 2010; Tang et al. 2014; Kiritchenko, Zhu, and Mohammad 2014; Cambria et al. 2014). For example, Kiritchenko et al. generated two tweet-specific sentiment lexicons based on words' associations with emoticons and hashtags containing sentiment word respectively (Kiritchenko, Zhu, and Mohammad 2014). Then a sentiment classification system was created which incorporates lexicon-related features, such as the number of positive and negative terms in a message, as well as other features into training and classification. Their system won the first place in SemEval-2013 competition. However, a word in different domains or contexts may convey different sentiments. For example, when describing CPU, the word "fast" is positive. Whereas, when describing battery, it usually conveys negative sentiment. For instance, "the battery runs out fast." The above lexicon based methods can not tackle this problem very well, because the same word is set to have the same sentiment polarity for different contexts.

In this paper, we propose a contextual knowledge regularization framework. Our framework can mine contextual knowledge of words from unlabeled data, and incorporate it as regularization terms into supervised learning framework to train a more accurate and robust model. Specifically, we establish two kinds of contextual knowledge from large scale short messages, i.e., the word-word association and word-sentiment association. The word-word association is the information indicating that two words may share similar sentiment. The word-sentiment association indicates the prior knowledge about the sentiment polarity of words. We propose to use a linear classification model to perform sen-

timent classification. We model the word-sentiment knowledge into a linear regularization term and model the word-word relation into a graph-guided fused lasso regularization term. We give an efficient optimization method based on ADMM (Boyd et al. 2011) to solve the regularized optimization problem. In addition, we propose an algorithm based on FISTA method (Beck and Teboulle 2009) to accelerate the most time-consuming component in the optimization procedure. We empirically validate our method with these two types of knowledge via extensive experiments on benchmark datasets. The experimental results show the effectiveness and efficiency of our method.

Contextual Knowledge

In this section we describe the two kinds of contextual knowledge: word-word and word-sentiment associations.

Word-Word Association

The assumption of this contextual knowledge is that if two words co-occur frequently in the same message, it is probable that they convey similar sentiment. For example, a tweet may be “Love love my iPhone 6! Sooooo beautiful!”. We can find many more cases that “love” and “beautiful” co-occur. As a consequence, we presume they share similar sentiment if they appear in a new short message. To more accurately perform the statistics, we compute the co-occurrence frequency by following rules. If a message contains adversative conjunctions such as “but” and “however,” then this microblog message will be split into different clauses in order to make sure that each clause conveys the consistent sentiment. If two words both appear in a clause, then their co-occurrence frequency increases by one. Formally, we use pointwise mutual information (PMI) as the measure of the sentiment similarity between a pair of words:

$$\text{PMI}(\text{word}_1, \text{word}_2) = \log_2 \frac{p(\text{word}_1, \text{word}_2)}{p(\text{word}_1)p(\text{word}_2)}, \quad (1)$$

where $p(\text{word}_1, \text{word}_2)$ represents the probability that word_1 co-occurs with word_2 , and $p(\text{word}_1)$ and $p(\text{word}_2)$ are the marginal probabilities of word_1 and word_2 . PMI score measures the statistical dependence degree between these two words. It has been used in sentiment analysis tasks such as sentiment lexicon construction (Turney and Littman 2002; Kaji and Kitsuregawa 2007; Kiritchenko, Zhu, and Mohammad 2014) and unsupervised classification of reviews (Turney 2002). Different from them, here we do not consider the emoticons but only compute the word level relatedness. This enables us to find more words that may share the same sentiment even though there is no emoticon to indicate the sentiment.

Word-Sentiment Association

An interesting phenomenon in microblogging services is that users tend to frequently use emoticons to express their emotions when posting microblogs. These emoticons can provide useful hints of sentiment. For instance, emoticons like “:)” and “:)” usually indicate positive sentiment and

emoticons such as “:(” tend to represent negative sentiment. Thus emoticons can be used as noisy sentiment labels, known as distant supervision. Some researchers have already tried this method to train sentiment classifiers and obtained certain accuracies (Go, Bhayani, and Huang 2009; Liu, Li, and Guo 2012). Motivated by naive Bayes classification, here we define the sentiment score of word i inferred from the distant supervision as:

$$\text{SentiScore}(\text{word}_i) = \log \left(\frac{\frac{n_i^+ + \alpha_0}{\sum_{j=1}^D n_j^+ + D \cdot \alpha_0}}{\frac{n_i^- + \alpha_0}{\sum_{j=1}^D n_j^- + D \cdot \alpha_0}} \right), \quad (2)$$

where n_i^+ and n_i^- are the frequencies that word i appears in positive and negative microblogs respectively. D is the length of vocabulary and $\alpha_0 > 0$ is a smoothing factor. According to Eq. (2), if a word has a higher probability to appear in positive microblogs rather than in negative microblogs, then its *SentiScore* will be larger than zero, which indicates that this word has a positive sentiment. Similarly, if a word is more likely to appear in negative microblogs, then its *SentiScore* will be less than zero and this word tends to convey a negative sentiment. More interestingly, the definition in Eq. (2) is in fact equivalent to the PMI-based score used by Kiritchenko et al. in (Kiritchenko, Zhu, and Mohammad 2014) when $\alpha_0 = 0$.

Contextual Knowledge Regularization

Contextual information has been proven to be useful for many NLP tasks (Subramanya, Petrov, and Pereira 2010; Das and Smith 2012). In this section, we introduce how to encode the contextual knowledge information into the regularized sentiment classification framework, and how to solve the corresponding optimization problem efficiently.

Notations

We denote $\mathbf{X} \in R^{N \times D}$ and $\mathbf{y} \in R^{N \times 1}$ as the training data. $\mathbf{x}_i \in R^{D \times 1}$ is the transpose of the i th row of \mathbf{X} , representing the feature vector of the i th sample, and $y_i \in \{+1, -1\}$ is the corresponding sentiment label. D is the dimension of the feature vector, i.e., the size of the vocabulary, and N is the number of training samples. Denote \mathbf{w} as the parameter vector of model, and $f(\mathbf{x}_i, y_i, \mathbf{w})$ as the loss of classifying \mathbf{x}_i into class y_i under the model parameter \mathbf{w} .

We first evaluate the PMI score on all pairs of words in the vocabulary. We only keep the PMI values $\text{PMI}(\text{word}_i, \text{word}_j)$ larger than threshold $\gamma_1 > 0$. Denote N_p as the number of remaining pairs. Then we construct a matrix $\mathbf{A} \in R^{N_p \times D}$. $A_{n,i} = 1$ and $A_{n,j} = -1$ if and only if $\text{PMI}(\text{word}_i, \text{word}_j) > \gamma_1$ and $\text{PMI}(\text{word}_i, \text{word}_j)$ ranks at the n th position. Otherwise, $A_{n,i} = 0$, $n = 1, \dots, N_p$ and $i = 1, \dots, D$. Since \mathbf{A} is highly sparse, we use sparse matrix to store it.

We use the vector $\mathbf{p} \in R^{D \times 1}$ to represent the knowledge of word-sentiment association. Given the sentiment score of word i calculated by Eq. (2), if it is larger than threshold $\gamma_2 > 0$, then $p_i = 1$. If it is less than $-\gamma_2$, then $p_i = -1$. Otherwise, $p_i = 0$. The introduction of thresholds is to filter out the contextual knowledge we are not very certain about.

Model

Given the training data and the contextual knowledge mined from unlabeled data, our goal is to train an accurate and robust sentiment classification model. The model proposed in this paper is as follow:

$$\arg \min_{\mathbf{w}} L = \sum_{i=1}^N f(\mathbf{x}_i, y_i, \mathbf{w}) - \alpha \mathbf{p}^T \mathbf{w} + \beta \|\mathbf{A}\mathbf{w}\|_1 + \lambda_1 \|\mathbf{w}\|_2^2 + \lambda_2 \|\mathbf{w}\|_1, \quad (3)$$

where $\alpha, \beta, \lambda_1, \lambda_2$ are non-negative regularization coefficients for word-sentiment associations, word-word associations, and model parameters. Our model is flexible to the selection of loss function f . f can be squared loss $(y_i - \mathbf{x}_i^T \mathbf{w})^2$, log loss $\log(1 + \exp(-y_i \mathbf{x}_i^T \mathbf{w}))$ and hinge loss $[1 - y_i \mathbf{x}_i^T \mathbf{w}]_+$. Here we use the L_1 -norm regularization of model parameter $\|\mathbf{w}\|_1$ because we believe that not all of the words will contribute to the final decision of sentiment classification. This can be regarded as feature selection for sentiment words. We also combine it with the L_2 -norm regularization term as elastic regularization, which is more stable in practice (Zou and Hastie 2003).

In Eq. (3), minimizing $-\mathbf{p}^T \mathbf{w}$ equals to minimizing $\|\mathbf{w} - \mathbf{p}\|_2^2$, because $\|\mathbf{w} - \mathbf{p}\|_2^2 = \|\mathbf{w}\|_2^2 + \|\mathbf{p}\|_2^2 - 2\mathbf{w}^T \mathbf{p}$, where $\|\mathbf{p}\|_2^2$ is a constant number and $\|\mathbf{w}\|_2^2$ can be merged into the L_2 -norm regularization term. By this formulation, we hope that the sentiment weight of a word learned here does not violate its original contextual sentiment polarity. Moreover, minimizing $\|\mathbf{A}\mathbf{w}\|_1$ equals to minimizing $\sum_{i,j} \text{s.t.} \{ \text{PMI}(\text{word}_i, \text{word}_j) > \gamma_1 \} |w_i - w_j|$. This means we constrain the learned sentiment weights of a pair of words should be similar if they are identified in the contextual knowledge. In this way, we penalize the original sentiment classification with these two regularization terms as soft constraints, which are controlled by α and β .

Optimization Method

Assuming the loss function f is convex as used in this paper, the optimization problem in Eq. (3) is also convex. However, it is still not easy to solve Eq. (3), even if the loss function f is smooth, due to the graph-guided fused lasso regularization term and L_1 -norm regularization term in the objective function. Here we propose an efficient algorithm based on the alternating direction method of multipliers (ADMM) (Boyd et al. 2011) to solve this optimization problem.

Before applying ADMM, we reformulate Eq. (3) as following optimization problem:

$$\begin{aligned} \arg \min_{\mathbf{w}, \mathbf{v}} L &= \sum_{i=1}^N f(\mathbf{x}_i, y_i, \mathbf{w}) - \alpha \mathbf{p}^T \mathbf{w} + \beta \|\mathbf{v}\|_1 \\ &\quad + \lambda_1 \|\mathbf{w}\|_2^2 + \lambda_2 \|\mathbf{w}\|_1, \\ \text{s.t.} \quad &\mathbf{v} = \mathbf{A}\mathbf{w}. \end{aligned} \quad (4)$$

As a method of multipliers, in ADMM, Eq. (4) is further formulated as an augmented Lagrangian problem:

$$\begin{aligned} L(\mathbf{w}, \mathbf{v}, \boldsymbol{\mu}) &= \sum_{i=1}^N f(\mathbf{x}_i, y_i, \mathbf{w}) - \alpha \mathbf{p}^T \mathbf{w} + \beta \|\mathbf{v}\|_1 \\ &\quad + \lambda_1 \|\mathbf{w}\|_2^2 + \lambda_2 \|\mathbf{w}\|_1 + \boldsymbol{\mu}^T (\mathbf{A}\mathbf{w} - \mathbf{v}) \\ &\quad + \frac{\rho}{2} \|\mathbf{A}\mathbf{w} - \mathbf{v}\|_2^2, \end{aligned} \quad (5)$$

where $\boldsymbol{\mu} \in R^{N_p \times 1}$ is the Lagrangian multipliers vector, and $\rho > 0$ is a penalty coefficient.

ADMM is an iterative optimization method. Unlike traditional multiplier methods where all the variables are optimized simultaneously in each iteration, in ADMM the variables \mathbf{w} , \mathbf{v} and $\boldsymbol{\mu}$ are optimized sequentially in an alternating manner, which allows the original problem to be decomposed into several easier sub-problems (Boyd et al. 2011). Denote $\mathbf{u} = \boldsymbol{\mu}/\rho$ as the scaled dual variable (Boyd et al. 2011), then in the t th iteration of ADMM, the variables \mathbf{w} , \mathbf{v} and \mathbf{u} are updated as follows.

Updating \mathbf{w}_{t+1} :

$$\mathbf{w}_{t+1} \leftarrow \arg \min_{\mathbf{w}} \sum_{i=1}^N f(\mathbf{x}_i, y_i, \mathbf{w}) - \alpha \mathbf{p}^T \mathbf{w} + \lambda_1 \|\mathbf{w}\|_2^2 + \lambda_2 \|\mathbf{w}\|_1 + \frac{\rho}{2} \|\mathbf{A}\mathbf{w} - \mathbf{v}_t + \mathbf{u}_t\|_2^2. \quad (6)$$

Updating \mathbf{v}_{t+1} :

$$\mathbf{v}_{t+1} \leftarrow \arg \min_{\mathbf{v}} \beta \|\mathbf{v}\|_1 + \frac{\rho}{2} \|\mathbf{A}\mathbf{w}_{t+1} - \mathbf{v} + \mathbf{u}_t\|_2^2. \quad (7)$$

Updating \mathbf{u}_{t+1} :

$$\mathbf{u}_{t+1} \leftarrow \mathbf{u}_t + \mathbf{A}\mathbf{w}_{t+1} - \mathbf{v}_{t+1}. \quad (8)$$

According to Eq. (8), updating \mathbf{u}_{t+1} is direct and trivial. The optimization problem in Eq. (7) can be solved using proximal algorithm (Parikh and Boyd 2013) and has an analytical solution:

$$\mathbf{v}_{t+1} = S_{\beta/\rho}(\mathbf{A}\mathbf{w}_{t+1} + \mathbf{u}_t), \quad (9)$$

where S is soft thresholding operator and is defined as $S_{\kappa}(a) = (a - \kappa)_+ - (-a - \kappa)_+$.

Unlike updating \mathbf{u}_{t+1} and \mathbf{v}_{t+1} , there is no analytical solution to the optimization problem in updating \mathbf{w}_{t+1} . It makes Eq. (6) the bottleneck of efficiency in the whole optimization problem. Thus, we should solve it in an efficient way.

When f is convex and smooth, such as squared loss and log loss, we propose here an accelerated algorithm based on FISTA (Beck and Teboulle 2009) to tackle Eq. (6). This algorithm keeps the advantage of low computational complexity as gradient method and subgradient method in each iteration, and at the same time it has a much faster convergence rate ($O(1/k^2)$) than gradient method ($O(1/k)$) and subgradient method ($O(1/\sqrt{k})$) (Beck and Teboulle 2009), where k is the number of iterations.

The core idea of FISTA is to use last two solutions to estimate current solution and iteratively update the approximate point \mathbf{z} and the search point \mathbf{s} . The search point \mathbf{s} is estimated by the linear combination of last two approximate points:

$$\mathbf{s}_{k+1} = \mathbf{z}_k + a_k(\mathbf{z}_k - \mathbf{z}_{k-1}), \quad (10)$$

where a_k is the combination coefficient at k th iteration.

Next we describe how to update the approximate point \mathbf{z}_{k+1} . First, we denote:

$$g(\mathbf{z}) = \sum_{i=1}^N f(\mathbf{x}_i, y_i, \mathbf{z}) - \alpha \mathbf{p}^T \mathbf{z} + \lambda_1 \|\mathbf{z}\|_2^2 + \frac{\rho}{2} \|\mathbf{A}\mathbf{z} - \mathbf{v}_t + \mathbf{u}_t\|_2^2. \quad (11)$$

Then the gradient of $g(\mathbf{z})$ is:

$$g'(\mathbf{z}) = \sum_{i=1}^N f'(\mathbf{x}_i, y_i, \mathbf{z}) - \alpha \mathbf{p} + 2\lambda_1 \mathbf{z} + \rho \mathbf{A}^T (\mathbf{A}\mathbf{z} - \mathbf{v}_t + \mathbf{u}_t). \quad (12)$$

When f is squared loss, we have $f'(\mathbf{x}_i, y_i, \mathbf{z}) = -(y_i - \mathbf{z}^T \mathbf{x}_i) \mathbf{x}_i$. When f is log loss, we have $f'(\mathbf{x}_i, y_i, \mathbf{z}) = -y_i \mathbf{x}_i / (1 + \exp(y_i \mathbf{z}^T \mathbf{x}_i))$.

Based on above derivations, the approximate point \mathbf{z}_{k+1} 's updating rule is:

$$\mathbf{z}_{k+1} = S_{\lambda_2/L_k} \left(\mathbf{s}_{k+1} - \frac{1}{L_k} g'(\mathbf{s}_{k+1}) \right), \quad (13)$$

where $\frac{1}{L_k}$ is the step size and L_k is selected according to following rule (Parikh and Boyd 2013):

$$g(\mathbf{z}_{k+1}) \leq g(\mathbf{s}_{k+1}) + g'(\mathbf{s}_{k+1})^T (\mathbf{z}_{k+1} - \mathbf{s}_{k+1}) + \frac{L_k}{2} \|\mathbf{z}_{k+1} - \mathbf{s}_{k+1}\|_2^2. \quad (14)$$

The complete accelerated algorithm for updating \mathbf{w}_{t+1} in Eq. (6) is summarized in Algorithm 1.

Algorithm 1 Accelerated algorithm for updating \mathbf{w}_{t+1} .

```

1: Input:  $\mathbf{X}, \mathbf{y}, \mathbf{A}, \mathbf{w}_t, \mathbf{v}_t, \mathbf{u}_t, \mathbf{p}, \alpha, \lambda_1, \lambda_2, \rho, \eta > 1, L_0$ .
2: Output:  $\mathbf{w}_{t+1}$ .
3: Initialize  $\mathbf{z}_1 = \mathbf{z}_0 = \mathbf{w}_t, k = 0, L = L_0$ .
4: while the convergence condition is not satisfied do
5:    $k = k + 1, a_k = \frac{k}{k+3}$ .
6:    $\mathbf{s}_{k+1} = \mathbf{z}_k + a_k (\mathbf{z}_k - \mathbf{z}_{k-1})$ .
7:    $g'(\mathbf{s}_{k+1}) = \sum_{i=1}^N f'(\mathbf{x}_i, y_i, \mathbf{s}_{k+1}) - \alpha \mathbf{p} + 2\lambda_1 \mathbf{s}_{k+1}$ 
      $+ \rho \mathbf{A}^T (\mathbf{A} \mathbf{s}_{k+1} - \mathbf{v}_t + \mathbf{u}_t)$ .
8:    $\mathbf{z}_{k+1} = S_{\lambda_2/L} (\mathbf{s}_{k+1} - \frac{1}{L} g'(\mathbf{s}_{k+1}))$ .
9:   while Eq. (14) doesn't hold do
10:     $L = \eta L$ .
11:     $\mathbf{z}_{k+1} = S_{\lambda_2/L} (\mathbf{s}_{k+1} - \frac{1}{L} g'(\mathbf{s}_{k+1}))$ .
12:   end while
13:    $\mathbf{z} = \mathbf{z}_{k+1}$ .
14: end while
15:  $\mathbf{w}_{t+1} = \mathbf{z}$ .
```

If the loss function f in Eq. (6) is not smooth, for example, when f is hinge loss function, then the accelerated algorithm in Algorithm 1 can not be applied. In this case, we propose to use subgradient method to solve the optimization problem in Eq. (6) (Shalev-Shwartz et al. 2011), whose convergence rate is $O(1/\sqrt{k})$.

Complexity Analysis

The convergence rate of the whole algorithm (the loop of updating \mathbf{w}_{t+1} , \mathbf{v}_{t+1} , and \mathbf{u}_{t+1}) is $O(1/T)$, where T is the number of iterations (Deng and Yin 2012; He and Yuan 2012). In other words, it takes $O(1/\epsilon_1)$ iterations to reach accuracy ϵ_1 . In each iteration, the time complexities of updating \mathbf{v}_{t+1} and \mathbf{u}_{t+1} are both $O(N_p)$. Assuming the time complexity of updating \mathbf{w}_{t+1} is $O(T_w)$, then time complexity of the whole algorithm is $O(\frac{T_w + N_p}{\epsilon_1})$.

When loss function f is squared loss or log loss, and Algorithm 1 is applied to update \mathbf{w}_{t+1} , then the convergence rate of Algorithm 1 is $O(1/k^2)$, where k is the iteration number (Beck and Teboulle 2009). Denote the desired accuracy for \mathbf{w}_{t+1} is ϵ_2 , then the number of iterations needed is $O(1/\sqrt{\epsilon_2})$. The major time complexity in each iteration of Algorithm 1 lies in updating search point (Step 6), gradient (Step 7) and approximate point (Step 8). The cost for

updating search point and approximate point is $O(D)$, and updating gradient needs $O(N \cdot D + N_p \cdot D)$ float-point operations. Thus the total time complexity of Algorithm 1 is $O(\frac{(N+N_p)D}{\sqrt{\epsilon_2}})$. In this case, the overall time complexity of the whole algorithm is $O(\frac{(N+N_p)D/\sqrt{\epsilon_2} + N_p}{\epsilon_1})$.

If loss function f is hinge loss and subgradient method is applied to update \mathbf{w}_{t+1} , then $O(1/\epsilon_2^2)$ iterations are needed to achieve accuracy of ϵ_2 . In each iteration of subgradient method, it needs $O(N \cdot D + N_p \cdot D)$ float-point operations. So the time complexity of updating \mathbf{w}_{t+1} here is $O(\frac{(N+N_p)D}{\epsilon_2^2})$ and the total time complexity of the whole algorithm is $O(\frac{(N+N_p)D/\epsilon_2^2 + N_p}{\epsilon_1})$.

Experiments

In this section, we present the experimental results on three Twitter sentiment classification benchmark datasets. The first dataset is Sanders Twitter sentiment dataset¹. This dataset contains 3,727 hand-labeled tweets related to four companies: Apple, Google, Twitter and Microsoft. The second dataset is the test data in Stanford sentiment corpus² (denoted as STS-manual) which was labeled manually. This dataset consists of 498 tweets in total and was obtained by crawling Twitter API using queries related to people, products, and companies. The third dataset is Twitter sentiment classification dataset provided by SemEval 2013 conference³ (denoted as SemEval). This dataset contains 8,258, 1,654 and 3,813 manually annotated tweets in the original training, development and test sets. However, some of them were non-existent now and we finally crawled 6,237, 974 and 2,465 tweets in training, development and test sets. In this paper, we focus on sentiment polarity classification. Only positive and negative tweets in these datasets were used in our experiments. Neutral tweets were filtered out. The detailed statistics of these datasets are shown in Table 1.

Table 1: Dataset Statistics

Dataset	Positive	Negative	Total
Sanders	570	654	1,224
STS-manual	182	177	359
SemEval-train	2,282	879	3,161
SemEval-dev	356	182	538
SemEval-test	988	333	1,321
STS-emoticon	800k	800k	1.6m

Several preprocessing steps were taken according to the suggestions in (Liu, Li, and Guo 2012). For example, stopwords were removed and all words were stemmed to terms and converted to lower cases. We didn't conduct any complex feature engineering and only unigram features were used in all of our experiments. Five-fold cross-validation was used for STS and Sanders datasets. For SemEval dataset, the original splitting was used. Parameters were

¹<http://www.sananalytics.com/lab/twitter-sentiment/>

²<http://help.sentiment140.com/for-students>

³<http://www.cs.york.ac.uk/semeval-2013/task2/>

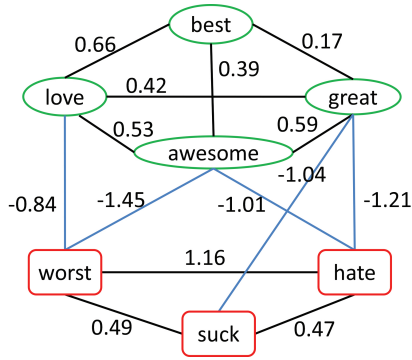


Figure 1: Word-word association examples. Words in green ellipses convey positive sentiment, and words in red rectangles have negative sentiment. The value on each line is the PMI score between the words connected by the line.

tuned on the validation sets. Each experiment was repeated 10 times independently and average results on test set were reported.

Contextual Knowledge Extraction

In order to extract the contextual knowledge from unlabeled data, we used a large dataset, i.e., the training data in Stanford sentiment corpus⁴ (denoted as STS-emoticon). It contains 1.6 million tweets crawled via Twitter API with emoticons as queries, among which half contain positive emoticons such as “:)” and half contain negative emoticons such as “:(”. Word-word and word-sentiment associations were extracted from this dataset.

Figure 1 illustrates the PMI scores between several representative words with clear sentiment polarity. We kept all the positive links and the negative links whose values are less than -0.8. Others were omitted for clarity. Figure 1 shows that words with the same sentiment have positive PMI scores and words with different sentiments share negative PMI scores. This indicates that words with the same sentiment are more likely to co-occur together, while the words with different sentiments are unlikely to appear together. When training a classification model, if the training data have a relatively large number of samples containing “hate” but few samples containing “worst,” then we can learn a more accurate weight for “worst” by using the weight of “hate” and the knowledge of association between “hate” and “worst.”

Table 2 illustrates the examples of word-sentiment associations with highest SentiScores (defined in Eq. (2)). We can see that words with high positive or negative SentiScores convey strong sentiment. In addition, we compared the sentiments of words extracted here with existing state-of-the-art sentiment lexicons, such as MPQA⁵ (Wilson, Wiebe, and Hoffmann 2005) and SentiWordNet⁶ (Esuli and Sebastiani 2006). Words whose absolute SentiScores greater than 0.5 were kept and others were filtered out. We de-

Table 2: Word-sentiment association examples.

Word	congratulate	kindly	feat	whee
SentiScore	2.59	2.26	2.09	2.06
Word	sadden	condole	toothache	sad
SentiScore	-3.33	-2.97	-2.87	-2.86

fine consistency between two lexicons as the ratio of shared words having the same sentiment label in both lexicons. Then the consistencies of our word-sentiment associations with MPQA and SentiWordNet are 79.2% and 79.9% respectively. These results indicate that word-sentiment associations extracted here match general sentiment lexicons quite well. At the same time, our word-sentiment associations contain a large number of popular informal words, such as “tnx” and “woohooo”, which express clear sentiment information but are not contained in traditional sentiment lexicons.

Performance

In this subsection, we compare our method with baseline methods on the three benchmark datasets. The methods to be compared are: 1) LS: Least squared method; 2) Log: Logistic regression; 3) SVM: Support vector machine; 4) NB: Multinomial naive Bayes with Laplace smoothing; 5) DistSup: Distant supervision method, where emoticons are used as sentiment labels (Go, Bhayani, and Huang 2009). Laplace smoothed naive Bayes was used as the classifier in DistSup here because it performs similarly with SVM and Maximum Entropy according to the original paper and is fast to train; 6) ESLAM: Emoticon smoothed language model, which linearly combines two naive Bayes classifiers, one built from manually labeled data and the other built using associations between words and emoticons (Liu, Li, and Guo 2012); 7) FeaLex: Extracting additional features, such as the numbers of positive and negative words in a message, using a tweet-specific sentiment lexicon, i.e., Sentiment140 (Kiritchenko, Zhu, and Mohammad 2014)⁷. SVM was used as classifier in FeaLex; 8) Contextual knowledge regularized least squared method (CK-LS), logistic regression (CK-Log) and support vector machine (CK-SVM), which are our methods under different types of loss function.

The results are shown in Table 3. We can see that our methods perform best on all the three datasets. The results of DistSup indicate that emoticons from Twitter are noisy sentiment labels and the performance of the sentiment classifier trained on emoticons directly is unsatisfactory. By combining the contextual knowledge mined from massive unlabeled data with the manually labeled data, our methods outperform their original counterparts significantly. Note that although ESLAM and FeaLex also use both manually labeled data and emoticons, our methods can still outperform them. This implies that our method using contextual knowledge as soft

⁴<http://help.sentiment140.com/for-students>

⁵<http://mpqa.cs.pitt.edu/>

⁶<http://sentiwordnet.isti.cnr.it/>

⁷Here we only incorporate unigram features and lexicon-related features, and do not incorporate other features such as POS tags, for the sake of fair comparison with other methods. Besides, features like POS tags have little influence on this method according to the original paper (Kiritchenko, Zhu, and Mohammad 2014).

Table 3: Accuracies of different methods.

	STS	Sanders	SemEval
LS	0.8228	0.8338	0.8191
Log	0.8218	0.8366	0.8229
SVM	0.7969	0.8218	0.7850
NB	0.8375	0.8207	0.8123
DistSup	0.7673	0.7297	0.7699
ESLAM	0.8698	0.8275	0.8251
FeaLex	0.8120	0.8350	0.8123
CK-LS	0.8642	0.8541	0.8388
CK-Log	0.8721	0.8579	0.8448
CK-SVM	0.8745	0.8530	0.8441

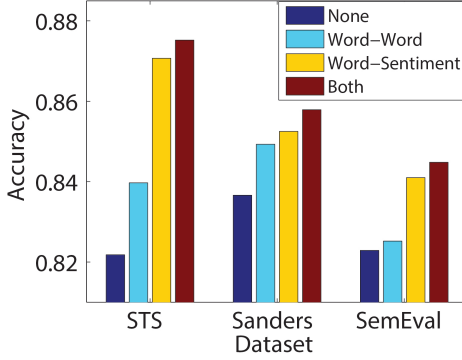


Figure 2: Effect of different kinds of contextual knowledge.

constraints is better than directly combining the manually labeled data with the emoticons as a new label set or extracting additional features from sentiment lexicons.

We also tested the contribution of different kinds of contextual knowledge to the performance of our method. The loss function used here is log loss and patterns of other loss functions are similar. The results are shown in Figure 2. We can see that both word-word and word-sentiment associations can help improve the classification. In addition, the performance of our method can be further improved when both kinds of contextual knowledge are used, which means that different kinds of contextual knowledge can cooperate with each other under the framework of our method.

Parameter Analysis

In this subsection, we explore the influence of the parameters. Here we concentrate on two important parameters, i.e., α and β , which control the importance of word-sentiment and word-word knowledge in the model. Here we take the results of CK-Log on SemEval dataset for example, which are shown in Figure 3. The patterns on other datasets or using other loss functions are similar. When α and β are small, the information of contextual knowledge is not fully used and the performance is improved when the parameters increase. However when α and β are too large, the information of contextual knowledge is overemphasized. The model will be overwhelmed by the contextual knowledge, which is not as accurate as the manually labeled data. Thus the per-

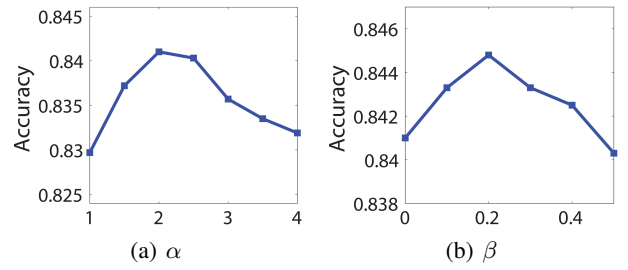


Figure 3: The influence of parameters.

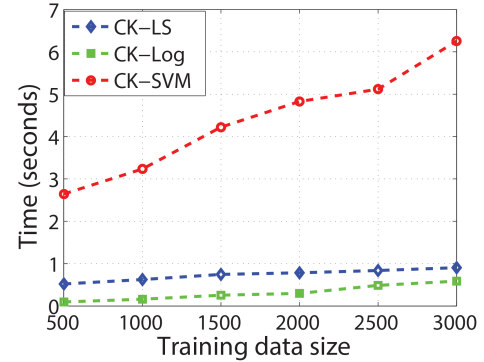


Figure 4: Time complexities of our methods.

formance will be harmed.

Efficiency

We conducted experiments to validate the time complexity of our method. Here we take the results on SemEval dataset for example, which are shown in Figure 4. We can see that the running time of all our methods is approximately linear with the data size, which validates the discussions in the Complexity Analysis section. In addition, CK-Log and CK-LS run much faster than CK-SVM, showing that the accelerated method in Algorithm 1 is useful for improving the time efficiency. Besides, CK-Log and CK-LS can finish training in 1 second on 3,000 samples, which is quite efficient.

Conclusion

This paper presents a way to incorporate the contextual knowledge mined from large scale unlabeled data into a regularization framework for microblog sentiment classification. We defined two kinds of contextual knowledge: word-word association and word-sentiment association. Word-word association indicates similar sentiment polarity between pairs of words, and word-sentiment association indicates the prior knowledge of the sentiment polarity of words. We also proposed an efficient algorithm to solve the optimization problem for the contextual knowledge regularization framework. Experimental results on three benchmark datasets show that our method can significantly improve the microblog sentiment classification performance.

Acknowledgments

This research is supported by National Science and Technology Major Project of China (NSTMP) under Grant No. 2012ZX03002015-003, 863 Program of China under Grant No. 2012AA011004 and Initiative Scientific Research Program of Tsinghua University under Grant No. 20111081023. Yangqiu Song gratefully acknowledges the support by the Army Research Laboratory (ARL) under agreement W911NF-09-2-0053, by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20155, and by DARPA under agreement number FA8750-13-2-0008. Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the view of the agencies.

References

- Baccianella, S.; Esuli, A.; and Sebastiani, F. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, 2200–2204.
- Beck, A., and Teboulle, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences* 2(1):183–202.
- Birmingham, A., and Smeaton, A. F. 2010. Classifying sentiment in microblogs: is brevity an advantage? In *CIKM*, 1833–1836.
- Bollen, J.; Mao, H.; and Pepe, A. 2011. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. In *ICWSM*, 17–21.
- Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; and Eckstein, J. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning* 3(1):1–122.
- Cambria, E.; Song, Y.; Wang, H.; and Howard, N. 2014. Semantic multidimensional scaling for open-domain sentiment analysis. *IEEE Intelligent Systems* 29(2):44–51.
- Dang, Y.; Zhang, Y.; and Chen, H. 2010. A lexicon-enhanced method for sentiment classification: An experiment on online product reviews. *IEEE Intelligent Systems* 25(4):46–53.
- Das, D., and Smith, N. A. 2012. Graph-based lexicon expansion with sparsity-inducing penalties. In *NAACL:HLT*, 677–687.
- Deng, W., and Yin, W. 2012. On the global and linear convergence of the generalized alternating direction method of multipliers. Technical report, DTIC Document.
- Esuli, A., and Sebastiani, F. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *LREC*, 417–422.
- Go, A.; Bhayani, R.; and Huang, L. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford* 1–12.
- He, B., and Yuan, X. 2012. On the $o(1/n)$ convergence rate of the douglas-rachford alternating direction method. *SIAM Journal on Numerical Analysis* 50(2):700–709.
- Hu, X.; Tang, L.; Tang, J.; and Liu, H. 2013. Exploiting social relations for sentiment analysis in microblogging. In *WSDM*, 537–546.
- Kaji, N., and Kitsuregawa, M. 2007. Building lexicon for sentiment analysis from massive collection of html documents. In *EMNLP-CoNLL*, 1075–1083.
- Kiritchenko, S.; Zhu, X.; and Mohammad, S. M. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research (JAIR)* 50:723–762.
- Liu, K.-L.; Li, W.-J.; and Guo, M. 2012. Emoticon smoothed language models for twitter sentiment analysis. In *AAAI*, 1678–1684.
- O’Connor, B.; Balasubramanyan, R.; Routledge, B. R.; and Smith, N. A. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *ICWSM*, 122–129.
- Parikh, N., and Boyd, S. 2013. Proximal algorithms. *Foundations and Trends in Optimization* 1(3):123–231.
- Shalev-Shwartz, S.; Singer, Y.; Srebro, N.; and Cotter, A. 2011. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming* 127(1):3–30.
- Subramanya, A.; Petrov, S.; and Pereira, F. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *EMNLP*, 167–176.
- Tang, D.; Wei, F.; Qin, B.; Zhou, M.; and Liu, T. 2014. Building large-scale twitter-specific sentiment lexicon : A representation learning approach. In *COLING*, 172–182.
- Turney, P., and Littman, M. L. 2002. Unsupervised learning of semantic orientation from a hundred-billion-word corpus. Technical report.
- Turney, P. D. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *ACL*, 417–424.
- Wilson, T.; Wiebe, J.; and Hoffmann, P. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/EMNLP*, 347–354.
- Wu, Y.; Liu, S.; Yan, K.; Liu, M.; and Wu, F. 2014. Opinion-flow: Visual analysis of opinion diffusion on social media. *IEEE Transactions on Visualization and Computer Graphics* 20(12):1763–1772.
- Zou, H., and Hastie, T. 2003. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67(2):301–320.