Improving Sentiment Classification Using Feature Highlighting and Feature Bagging

Lin Dai  
Computer Science School  
Beijing Institute of Technology  
Beijing 100081, China  
daili@bit.edu.cn

Hechun Chen  
Baidu Inc.  
No. 10, Shandong 10th Street  
Beijing 100085, China  
chenhechun@baidu.com

Xiaomei Li  
Beijing Electronic Science and Technology Institute  
7, Fengshan Road, Fengtai District  
Beijing 100070, China  
lixueming@besti.edu.cn

Abstract—Sentiment classification is an important data mining task. Previous researches tried various machine learning techniques while didn’t make fully use of the difference among features. This paper proposes a novel method for improving sentiment classification by fully exploring the different contribution of features. The method consists of two parts. First, we highlight sentimental features by increasing their weight. Second, we use bagging to construct multiple classifiers on different feature spaces and combine them into an aggregating classifier. Extensive experiments show that the method can evidently improve the performance of sentiment classification.

Keywords—Sentiment analysis; Feature highlighting; Feature bagging.

I. INTRODUCTION

With the rapid development of internet, more and more kinds of online information are available. In these information resources, there are abundant of subjective comments and views, for example, comments for electronic products, cars, movies, or reviews for some events or policies. It is valuable to decide whether the comments are positive or negative, if the reviews are supportive or oppositive, and if a product is recommendable or not? These requirement elicited the research topic of sentiment analysis. In some literatures, sentiment analysis is also referred as opinion mining. Its primary objective is to predict the sentimental polarity of given context by identifying sentimental constituents like sentimental words, subjective sentences, etc. Sentiment analysis examines a wide variety of texts, from sentences to documents, such as forum comments, web pages, blog articles, product reviews, and movie reviews, etc. Pang et al [2] made a comprehensive review of sentiment analysis and opinion mining.

Nowadays, sentiment analysis is deployed very extensively in internet settings, because it can, to some extent, help to manage large scale of information and to locate interesting information. Particularly, sentiment analysis is very helpful to analyze consumers’ feedbacks and consuming tendency. One example is in recommending systems, sentiment analysis helps to automatically classify online feedbacks of products or services, and select recommendable ones for consumers. Pang et al [1] introduced review mining to re-rank search engine results. Their method substantially improves the initial results produced by the Yahoo! search engine.

Previous researches applied various machine learning algorithms on sentiment classification while didn’t make fully use of the different contribution of features. But for sentiment classification, it is intuitive that feature selection and weighting scheme plays a critical role. Based on this premise, this paper, we are to try to improve sentiment classification by: 1, feature highlighting, i.e., highlight labeled sentimental features by elevating their weight. 2, feature bagging, i.e., train sub-classifiers on randomly selected feature spaces and then combine them into a final classifier by voting. Experimental results indicate the effectiveness of feature highlighting and feature bagging.

The rest of this paper is organized as follows. Section 2 introduces related works. Section 3 elaborates on our method. Section 4 is about experiments and discussion before concluding this paper at the last section.

II. RELATED WORK

The task of sentiment classification is classifying the polarity of the expressed opinion [3], which can be positive or negative, supportive or not, and etc. Various machine learning methods have been introduced into this task. For example, Turney [4] utilized web search engine to compute semantic tendency. Hu et al [5] tried to classify word into opinion and non-opinion terms. [18] explored word relations for sentiment classification.

Pang et al [6] applied three different methods, i.e., Support Vector Machines (SVMs), Naïve Bayes and Maximum Entropy, to conduct a comparison experiment, in which SVMs yielded the best result. Besides, for feature weighting, they found that in sentiment classification, Binary (presence/absence of words) is better than Term Frequency (TF). They collected 1400 film comments as experiment data set, on which the highest accuracy reached 82.9%. Later, Pang et al [7] achieved even better result by applying artificial filtration to get rid of objective sentences and select subjective sentences for learning, which addressed the importance of subjective features and sentences for sentiment classification.

Recently, researches were trying novel techniques to cope

Schapire [12] proposed the first boosting algorithm. Later, Freund [13] proposed AdaBoost algorithm which is more efficient than boosting. Breiman [14] proposed bagging algorithm which utilizes the instability of classifiers. To apply integrating learning algorithms to sentiment classification, Li et al [15] generated different classifiers on feature sets with different POS, then combine them into a final classifier. It was found that the final classifier exhibited improved accuracy, with an improvement of 2.56% over the best individual classifier. Similarly, Tsutsumi et al[16] integrated three kinds of classifiers and got some improvement. [19] used POS-based ensemble model for cross-domain sentiment classification.

III. Algorithm

We are trying to improve the performance of sentiment classifiers by exploring the different contribution of features. To do this, two strategies are proposed, i.e., feature highlighting and feature bagging, as follows.

A. Strategy 1: Feature Highlighting

With vector space model (VSM), a document d is considered as a set of features and corresponding weights, i.e., a vector \( \vec{d} = (w_1, w_2, ..., w_n) \), where \( w_i \) is the weight of feature \( f_i \). \( \vec{d} \) is called the representation vector of document d.

There are several means to compute feature’s weight, such as Binary (presence/absence of words), TF, TF-IDF and so on. In traditional topic-oriented text classification, TF-IDF has been proven better than Binary and TF. But for sentiment classification, Binary turns out to be more effective than TF and TF-IDF[6]. This is partially because the sentimental polarity does not merely depend on topic keywords, but also depend on specific sentimental words, often adjective, such as "good", "perfect", "wasteful", etc. Usually, these sentimental words rarely appear repeatedly due to language custom and wide range of selection. So TF and TF-IDF can’t work well to distinguish sentimental attributes because they can introduce “irrelevant” features. With binary representation, a document is represented as:

\[
\vec{d} = (I(f_1), I(f_2), ..., I(f_m))
\]

where \( I(\cdot) \) is a indicator function. If \( f_i \) appears in d, \( I(f_i) = 1 \), otherwise \( I(f_i) = 0 \).

However, in binary representation, the features are equally treated. It does not highlight the critical effect of sentimental constituents (such as sentimental words). For example, given sentence “The cumulative effect of the movie is repulsive and depressing”. We confirm this sentence expressing negative sentiment according to the sentimental words repulsive and depressing. If binary representation is deployed, obviously, all the features are equally important. In other words, it doesn’t reflect the dominant effect of the features. It is acknowledged that the greater the weight of a feature is, the more indicative it is for the article’s class. Therefore, to highlight the impact of sentimental words, we elevate the weight of sentimental words, which may be labeled ahead of time. After highlighting, a feature vector is:

\[
\vec{d}' = (w(f_1), w(f_2), ..., w(f_m))
\]

where \( w(f_i) \) is the weight of feature \( f_i \). If \( f_i \) appears in d and sentimental word list, \( w(f_i) = k \). If \( f_i \) appears in d and not in sentimental word list, \( w(f_i) = 1 \). Otherwise \( w(f_i) = 0 \). Here, \( k > 1 \) is the highlighting factor which aims to elevate the weight of sentimental words. As the above example is concerned, if the words repulsive and depressing are in sentimental word list, their weight will be lifted, so that their contribution to sentiment classification is amplified. In practice, \( \vec{d}' \) should be L2 normalized.

B. Strategy 2: Feature Bagging

Breiman [14] proposed the bagging algorithm. Given a learning set of L consists of data \( \{(x_n, y_n), n = 1, ..., N\} \) where y’s are class labels. Take repeated bootstrap samples \( \{L^B\} \) from L, each consisting of N cases, drawn at random and with replacement, from L. Each \( (x_n, y_n) \) may appear repeated times or not at all in any single \( L^B \). Bagging trains a sequence of sub-classifiers \( \{\psi(x, L^B)\} \) from \( \{L^B\} \). The aggregated classifier can be formed by voting. If \( \psi(x, L^B) \) predicts a class \( j \in \{1, ..., J\} \), let \( N_j = \#\{k: \psi(x, L^B) = j\} \), then \( \psi_A(x) = \arg \max_j N_j \). Here the subscript A in \( \psi_A \) denotes aggregation.

A critical factor in whether bagging will improve accuracy is the stability of the procedure for constructing \( \psi \). Improvement will occur for unstable procedures where a small change in L can result in large changes in \( \psi \). Instability was studied in Breiman [14], where it was pointed out that neural nets, classification trees were unstable, while kNN were stable.

But using the unstable classifiers are not the only way to reach improvement. We argue that any factor that causes classifier to be unstable can used to improve original classifier. Besides unstable classifiers, stable ones also can be used as the base predictors in bagging. What we need to do is to perturb the procedure of constructing classifiers to make them unstable.

It is well known that the selection of features is a critical factor that will “change” classifier’s performance. That is to say, classifiers will be unstable if they are learned on elaborately generated feature spaces. This instability is
right the requirement of improvement for bagging. We call bagging on feature sets as feature bagging. The basic idea of feature bagging strategy is to implement bagging through varying feature space: learn a sequence of classifiers on different feature spaces, then form aggregated classifiers by voting.

Formally, given learning set $L$ and a feature space $F$. Take repeated sub-feature spaces $\{F^B\}$ from $F$ with a proportion, drawn at random and without replacement. Then train a sequence of sub-classifiers $\{\varphi(x, L, F^B)\}$ on $\{F^B\}$. The aggregated classifier is $\varphi_A(x) = \arg \max_j N_j$, $N_j = \# \{ k | \varphi(x, L, F^B) = j \}$.

C. Why do the strategies work?

As Strategy 1 is concerned, artificially increasing the weight of sentimental features is equal to form artificial learning samples in which the reoccurring possibility of sentimental words increases. These sentimental words are informative to sentiment classification and appropriately increasing their TF will not change document’s sentiment attribute. So the result of highlighting sentimental features is highlighting documents’s sentiment attribute and thus improve classifiers’ performance.

The explanation for Strategy 2 is more complicated.

**Axiom 1.** Invariability of Category Attribute. For text classification including sentiment classification, given a document, any part from it with an enough large proportion remains its category attribute.

For example, if we randomly choose one third or one half from a document to make new documents, the category attribute of the new documents should be as same as the original document. However, when the extraction proportion is too small, such as just one or few words, it is possible that the extracted document can not represent the original document’s category because of the lost of useful features.

**Definition 1. Partition set.** Given a document set, for each document in it, extract words from it with a proportion to form new documents, thus form a new document set. We call this new document set a partition set of the original set.

Based on Axiom1, we can get partition sets from a learning document set. For classification task, obviously, partition sets are similar because the category attribute of each document in them does not vary.

**Presumption 1.** Given a labeled document set, if it is partitioned with a proper proportion, classifiers learned from its partition sets is comparable with the classifier learned from itself.

We examine this presumption experimentally. As a validation experiment, a task of binary text categorization is examined. We got web pages of category Automobile and Education, with size of 700 respectively. 100 documents of each category are selected at random as learning set and the remains as testing set. Partition sets are formed by extracting different words with certain proportion from each document, without repetition. SVMs are trained on partition sets (libsvm [17] is used). For each proportion, experiments are performed 10 times and the average accuracy is recorded, as shown in Figure 1.

From Figure 1 we can see that when the proportion falls in [0.5, 1.0], the accuracy of SVMs are very similar without falling. And the performance falls evidently till the proportion is less than 0.25. While when the proportion falls in [0.25, 0.5] the performance fluctuates. The results prove this presumption.

**Axiom 2.** For learning classifier, if binary representation (presence/absence of words) is adopted, randomly sampling on feature space with fixed learning set is equal to randomly forming partition set with fixed feature set.

In other words, for a given document, the effect of using a randomly generated sub-feature space to represent it, is equal to using randomly words from it to represent it with original feature space. And vice versa. Because for any parts of a given document, we can always find a sub feature space in which the representation of the original document is identical to the representation of the extracted parts in original feature space.

**Theorem 1.** Feature bagging with proper proportion can yield improved performance.

Proof: According to Axiom 1 and Axiom 2, we know that feature bagging is equal to bagging on partitioned sets. From presumption 1, we know that classifiers’ performance on partitioned sets may be similar to original learning set, or may fluctuate on different feature space. It is known that bagging on learning set can improve performance, which is proved in literature [14]. So, if classifiers’ performance varies on different partitioned sets, which reflects the different contribution of features, without loosing too much accuracy, classifier’s performance can be improved.

Apparently, there are two key factors in whether feature bagging can improve accuracy. One is classifier’s instability on different partition sets and the other is whether the performance of classifier deteriorates too much. There is
a cross-over point at which \( \varphi_A(x,F^B,L) \) stop improving \( \varphi(x,L) \) and does worse.

IV. Evaluation

A. Set-up

Pang et al compared three classification algorithms in literature \[6\]. When unigrams and Binary were used to represent documents, SVMs is better than Naïve Bayes and Maximum Entropy by 2%~3%. Based on their results, in our experiments, SVMs are used as the basic classifier.

We use the movie review dataset as literature \[2\] did. This dataset contains total of 1400 film comments, i.e., 700 positive comments and 700 negative comments. The dataset is firstly pre-processed, including feature fltering and negative words processing, as follows. (1) Filter features according to their frequency. We exclude the words which appear less than 4 times in this dataset. (2) Add the prefix “NOT_” to every word after a negation word (“not”, “no”, “isn’t”, etc.). They will be seemed as sentimental words in the training and classifying stage. In pre-processing, no stemming is performed nor stoplist is applied, since they have little impact to the final performance.

After pre-precession steps we got original feature space of size more than 15000. We use unigrams and VSM to represent comments. L2-normalization is performed on feature vectors except Binary.

Three-fold cross-validation is used to get average performance of algorithms. The whole dataset is randomly and equally divided into three portions. In each iteration, two portions are used as training data and the remaining one as testing data. The average accuracy of the three iterations is recorded as performance of algorithms. The accuracy is calculated as:

\[
\text{accuracy} = \frac{N(d')}{N(d)}
\]

\[ \text{(3)} \]

where \( N(d) \) is the total number of test comments and \( N(d') \) is the number of comments which are classified properly.

In order to label sentimental words, OpinionFinder’s subjectivity lexicon \[7\] is introduced. It consists of over 8000 subjectivity clues which are compiled from several sources. Words in this lexicon are labeled by four types of sentiment: positive, negative, both, and neutral. And according to sentimental strength, words in this lexicon are classified into two types: weaksubj and strongsubj. We choose positive and negative words as sentimental features. The total number of the sentimental features is 7630 and the quantitative distribution is shown in Table I.

As the sentimental strength of strongsubj words is different from that of weaksubj words, we give them different highlighting factor, \( k_s \) and \( k_u \), respectively, which are to be determined by experiments.

<table>
<thead>
<tr>
<th>#strongsubj</th>
<th>#weaksubj</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>1589</td>
</tr>
<tr>
<td>Negative</td>
<td>3737</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weighting</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF</td>
<td>76.68%</td>
</tr>
<tr>
<td>Binary</td>
<td>81.26%</td>
</tr>
</tbody>
</table>

Table I

STATISTIC FEATURES OF SENTIMENTAL WORD LIST.

B. Experimental results

1) Single SVMs (Baseline): First, we use single Support Vector Machines to train and predict the movie comments. Libsvm library \[17\] is used to implement SVMs. For all SVMs we use linear kernel and with parameter \( c=2.0 \) and \( \gamma = 2.0 \). The TF and Binary weighting scheme are evaluated. The experimental results are listed in Table II.

From Table II we can see that the accuracy of Binary is obviously higher than that of TF, which verifies the difference between sentiment classification and traditional text classification. In the following experiments, the performance of the Binary is used as baseline.

2) Sentimental Feature Highlighting: In this experiment, we experimentally check the effect of sentimental feature highlighting and the optimal parameter \( k_s \) and \( k_u \). The graph of accuracy against \( k_s \) and \( k_u \) is show in Figure 2 (a) and (b), respectively.

From Figure 2 we can see that with appropriate weight promotion of sentimental words, classifier’s accuracy will be increased to some extent. But the performance of classifier will deteriorate if sentimental words are highlighted too much. The optimal value for \( k_s \) and \( k_u \) is 3 and 2.2 respectively. The higher value of \( k_s \) verified the higher indicative effect of strongsubj words than weaksubj words. In the following experiments, the optimal values will be used.

3) Feature bagging: This experiment implements feature bagging in which sub-classifiers are trained on different feature spaces. The objective is to verify the importance of feature selection for sentiment classification and the performance lifting benefits from boosting learning on randomly feature sampling. Through experiments we found that when \( T \) is greater than 50, the performance of final classifiers become stable. So we set \( T=50 \) in this experiment. We train \( T \) sub-SVMs on different sub-feature spaces which are randomly extracted from original feature space with proportion \( \lambda \). Then, the sub-SVMs are integrated by voting mechanism.
The performance of final classifiers with different $\lambda$ are shown in Figure 3.

When $\lambda$ is 1, the algorithm is equal to original SVMs. With the decrease of $\lambda$ the accuracy increase correspondingly. The accuracy reaches its highest value when $\lambda$ is 0.3. After that, when $\lambda$ is further decreased, the accuracy decreases correspondingly. After integrating, the final classifier outperforms the baseline by 2.2%. This reflects the advantage of bagging idea.

Table III gives out the accuracy of several random selected sub-classifiers when $\lambda$ is 0.3. It is worth noticing that for each sub-classifier on sub-feature space, its accuracy is worse than the baseline. This is because of that in sub-feature space, an abound of informative features are skipped which make original classifiers become weak classifiers. This satisfies the requirement for weak classifiers of bagging boosting algorithm.

Another important observation from table III is that even with $\lambda = 0.3$, the accuracy of sub-classifier on different sub-feature space varies obviously, with variance of 6.19. This observation proves our basis premise: sentiment classification greatly depends on feature selection, and, feature sampling with relatively smaller ratio can satisfies the instability requirement of sub-classifiers of bagging boosting.

4) Combination of feature bagging and feature highlighting: In this experiment, the above two strategies are combined to further improve the classifier. On the basis of sentimental feature weight lifting, again, boosting learning on feature bagging is deployed. With the optimal parameters got from previous experiments, we got the final accuracy of 84.63%. For comparison convenience, Table IV lists the accuracy of the four methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single SVMs</td>
<td>81.26%</td>
</tr>
<tr>
<td>Feature Highlighting</td>
<td>83.27%</td>
</tr>
<tr>
<td>Feature Bagging</td>
<td>83.52%</td>
</tr>
<tr>
<td>Combination</td>
<td>84.63%</td>
</tr>
</tbody>
</table>

Table IV

RESULT OF FOUR METHODS: SINGLE SVMs, FEATURE HIGHLIGHTING, FEATURE BAGGING AND COMBINATION.

From Table IV we can see that, compared with baseline, sentimental feature highlighting technique outperforms the single SVMs by 2.01%. And feature bagging technique improves the accuracy by 2.36%. The combination of the two techniques yields accuracy of 3.37% higher than the
baseline. This result shows the overall performance of the proposed method.

V. CONCLUSIONS AND FUTURE WORK

In this paper, the impact of features to sentiment classification was studied. The objective is to minimize the interference from unrelated features while highlighting the contribution of sentimental words. First, sentimental words are highlighted by weight lifting. Second, multiple classifiers are learned from randomly generated sub-feature spaces. Then the final classifier is set up from these sub-classifiers by voting. We named the two strategies as feature highlighting and feature bagging. Based on experimental outcomes, we can conclude that:

- Sentimental words play a critical role in determining the sentiment polarity of a document. To increase accuracy of classification, it is important to increase the weight of sentiment words although they are not the only determinant.
- Aggregating learning method based on randomly feature sampling can increase the accuracy of classifiers, behaving like traditional bagging.
- The improvement of the above two techniques is stable, the combination of them could substantially increase the accuracy.

Although it is found that highlighting sentimental words was proved to be very useful in sentiment classification, it is arbitrary to give them the same highlighting factor. We are planning to generate fine weight scheme by machine learning techniques to further improving classifiers.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (60803050, 61132009) and BIT Team of Innovation.

REFERENCES


