Sentiment Classification Using Machine Learning Techniques with Syntax Features

Huang Zou
School of Software Engineering
Shanghai Jiao Tong University
Shanghai, China
zhstevenash@sjtu.edu.cn

Xinhuai Tang
School of Software Engineering
Shanghai Jiao Tong University
Shanghai, China
tang-xh@cs.sjtu.edu.cn

Bin Xie
The thirty-second Research Institute
China Electronic Technology Group Corporation
Shanghai, China
xiebin_sh@163.com

Bing Liu
Shijiazhuang Institute
Engineering College
Shijiazhuang, China
liubbb@163.com

***Abstract*—Sentiment classification has adopted machine learning techniques to improve its precision and efficiency. However, the features are always produced by basic words-bag methods without much consideration for words' syntactic properties, which could play an important role in the judgment of sentiment meanings. To remedy this, we firstly generate syntax trees of the sentences, with the analysis of syntactic features of the sentences. Then we introduce multiple sentiment features into the basic words-bag features. Such features were trained on movie reviews as data, with machine learning methods (Naive Bayes and support vector machines). The features and factors introduced by syntax tree were examined to generate a more accurate solution for sentiment classification.**

Keywords—Sentiment classification; Syntax tree; POS features; Machine learning

I. INTRODUCTION

Sentiment classification is a useful technique to analyze the huge amount text on web. Traditionally, unlike the rating information, natural text can not be properly used in analysis. However, with the help of sentiment classification, which is usually conducted on the most normal words (such as movie and book reviews), more information about users and items can be provided.

One typical approach for sentiment classification is to analyze the words-bag features of text with supervised machine learning algorithms [1]. In such approach, all the words, which can also be filtered, form a words-bag vector. For each text, the appearance of words in such vector will be represented as the feature of the text. Besides, n-gram, negation-tags and POS tags are often used to optimize it. N-gram and negation-tags are supposed to improve the precision of the algorithm while POS tags can rule out the ambiguity different POS of one word bring.

Another approach to build a sentiment classifier is based on sentence syntax tree [2]. The sentences are parsed to construct a syntax tree to represent the relationship between words. Then the model, or pattern of sentiment classifier could be generated using the polarity of words, their POS features and their syntax relations.

Surely syntax features can paly a great role in the sentiment behind the sentence. For example, we would normally think the words in the main clauses would have more sentiment significance than those in the subordinate clauses, for people would always express their emotion directly, especially in the comments of books and movies, etc. However, such features have been seldom used in the words-bag classifiers. So we incorporate the syntax features into the implementation of words-bag sentiment classifier. Moreover, we proved that the words dependencies can also be reliable improvement for sentiment classification.

We use the Stanford Parser to generate syntax tree, words dependencies as well as the POS tags for sentences and words. Then we go through the trees for different syntax features – for example, the kind of the syntactic part (noun phrase, verb phrase, simple declarative clause, etc.).

Our unique features mainly come from three aspects. 1) The location of the word – whether it is in the main clause or the subordinate clause, can make a difference in the process of sentiment classification. 2) Words dependencies, or grammatical relationship between words also reveal information about emotions. 3) Lastly, as many pre-works have mentioned, POS of words are employed to build our classifier.

We use the data published by Pang and Lee [1] to experiment our method. The text was firstly used as input for Syntax Parser to generate syntax tree, which contains POS and syntax features. Then such features were used as the final vector features of the text. The experiment was conducted on two machine learning methods -- Naive Bayes and support vector machines (SVM).

The rest of this paper is organized as follows: Section 2 describes the related work of sentiment classification; Section 3 focuses on the Syntax Tree and its generation; Section 4 mainly talks about our methods to generate the features with the combination of POS and syntax tags; Section 5 shows the experiments, the results, and our conclusion.

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.44

TABLE I. Precision of Stanford Parser in the generated syntax tree about POS tags and Syntactic tags of words in different clauses.

	All clauses	*Main clauses*	*Main clauses + 1st-level subordinate sentences*
POS tags	95.3%	95.7%	94.9%
Syntactic tags	91.4%	96.4%	93.5%

II. Related Works.

Words-bag is a widely used method to conduct sentiment classification along with machine learning methods. Pang and Lee [1] select the words in movie reviews as features and examine those features through different machine learning methods. Also, they explore different ways to generate words-bag and words feature vectors. Dave, Lawrence and Pennock [3] also use machine learning methods to explore sentiment classification. However, they select top words according to their generated points instead of using all the words. Tony Mullen and Nigel Collier [4] use SVM to analyze sentiment orientation of words as well as topic-oriented and artist-oriented information. Pak and Paroubek [5] develop a sentiment classifier for twitter data using words-bag method relying on features from twitter corpus, which shows the application of sentiment analysis in social network.

Syntax trees are also developed by many in order to extract more internal relationship between words. Maximum entropy models are used by Adwait Ratnaparkhi [6] to parse syntax trees. The main target of such method is to find the patterns behind syntax tree. Wilson, Wiebe and Hoffman [7] develop some sentence features and structure features and set some rules to judge prior polarity. Zhan, Li and Zhu [8] also focuse on parsing the syntax tree with rules and patterns, which greatly improves the accuracy.

Some other models are also used to classify sentiment. Nakagawa, Inui and Kurohashi [9] make use of CRF with hidden variables to generate the dependency of sentiment on words, and then the sentiment of sentences. Duric and Song [10] also construct HMM model to analyze the content and syntax of sentences.

Domain issue is also a problem for sentiment classification. The model or classifier trained in one domain often does not perform well on another domain. To remedy this problem, Aue and Gamon [11] try to limit text features to those observed in the target domain. Yang, Callan and Si [12] rely on knowledge transfer with opinion word dictionary. Dunning [13] develops a measure based on likelihood ratios to analyze cross-domain text.

There're other methods for sentiment classification. Whitelaw, Garg and Argamon [14] constructs lexicon structures for words and their sentiments, which leads to a rule for classification. Prabowo and Thelwall [15] combine different classifiers to ensure taking the advantages of every classifier. When one classifier fails to return a right sentiment, the algorithm can employ another classifier to accomplish the task. Turney and Littman [16] determined the words' similarity with NEAR operation on web searches and build classifier based on the words polarity already known. Such method has been widely used.

III. Prepare Your Paper Before Styling

To accomplish our idea to make use of the syntactic features, we need to generate syntax trees firstly. We use the Stanford Parser to help us finish such target. Below is the sample of one sentence and its generated syntax tree from the website of Stanford Parser in Fig. I:

"The strongest rain ever recorded in India shut down the financial hub of Mumbai, snapped communication lines, closed airports and forced thousands of people to sleep in their offices or walk home during the night, officials said today."

```
(ROOT
  (S
    (S
      (NP
        (NP (DT The) (JJS strongest) (NN rain))
        (VP
          (ADVP (RB ever))
          (VBN recorded)
          (PP (IN in)
            (NP (NNP India)))))
      (VP
        (VP (VBD shut)
          (PRT (RP down))
          (NP
            (NP (DT the) (JJ financial) (NN hub))
            (PP (IN of)
              (NP (NNP Mumbai)))))
        (, ,)
        (VP (VBD snapped)
          (NP (NN communication) (NNS lines)))
        (, ,)
        (VP (VBD closed)
          (NP (NNS airports)))
        (CC and)
        (VP (VBD forced)
          (NP
            (NP (NNS thousands))
            (PP (IN of)
              (NP (NNS people))))
          (S
            (VP (TO to)
              (VP
                (VP (VB sleep)
                  (PP (IN in)
                    (NP (PRP$ their) (NNS offices))))
                (CC or)
                (VP (VB walk)
                  (NP (NN home))
                  (PP (IN during)
                    (NP (DT the) (NN night))))))))))
    (, ,)
    (NP (NNS officials))
    (VP (VBD said)
      (NP-TMP (NN today)))
    (. .)))
```

Fig. I. Sample of one generated syntax tree by Stanford Parser.

Clearly, with the help of Stanford Parser, we can easily get a syntax tree with POS and syntactic information. To ensure the precision of Stanford Parser, we collect 500 sentences and judge the POS and syntactic tags of every word manually to examine whether it works efficiently. Taking out empirical situation into account, details in subordinate clauses is not the same important as the information in main clauses. So we conduct the same experiment after we rule out the subordinate

clauses which is too deep. The results are shown in Table I. From the results we can see that Stanford Parser has a high precision in POS and syntactic tags determination. Also, the depth of subordinate clause has an influence on the precision of syntactic tags. But such difference is little as for the POS tags.

IV. Feature Extraction

Firstly, we assume that we have applied Stanford Parser to the dataset text and generate syntax POS tags for every word, and syntactic tags for words and sentences successfully. Then we can focus on how to extract features from such syntax trees. We try to capture features from the following aspects.

A. Main Clauses vs. Subornidate Clauses.

Intuitively, we would think that words in the main clauses have a difference with words in subordinate clauses as for their underlying sentiment. People always express their overall emotions directly in the main clauses, especially in those reviews on books and movies, while the subordinate clauses are mostly used to provide more detailed information about the subjects.

Taking such feature into consideration, we go through syntax trees to identify whether the words are in the main clauses or subordinate clauses. According to the syntactic tags from Peen Treebank, we can set "SBAR" and "SBARQ" to be the sign of a new subordinate clause. Thus the depth of every sentence can be known. In our case, for every new word "W" in the text, if it is in the main clause, we will add "W-m" to the words-bag collection; if it is in the subordinate clauses, then we will add "W-s" to the words-bag collection. Through such method, we can generate a new kind of words-bag compared with the traditional words-bag method.

B. Main Clauses vs. Subornidate Clauses.

Words dependencies give a description of the grammatical relationships in the words in a sentence. This can be quite useful when applied in generating the bigrams for words-bag. Normally, bigram words-bag are generated by selecting every two adjacent words into words-bag collection. The primary motivation for bigram is to identify logistic relation between words, which unigram method can't present. However, the bigram method now is not efficient, because it ignores all the underlying relationship between those words which are not adjacent. With words dependencies, we can remedy such problem because real logistic relationship can be found in words dependencies, even when words are not adjacent.

We use the words dependencies result generated along with syntax tree. For example, below is one sample sentence and its words dependencies:

"As an audience, we're also given a situation where two wonderfully talented actors are thrown into a movie, and we'd like to see if one will dominate the film. Both provide some pretty good entertainment."

nmod(given-8, as-1)
det(audience-3, an-2)
dep(as-1, audience-3)
nsubjpass(given-8, we-5)
auxpass(given-8, 're-6)
advmod(given-8, also-7)
root(ROOT-0, given-8)
det(situation-10, a-9)
dobj(given-8, situation-10)
advmod(thrown-17, where-11)
nummod(actors-15, two-12)
amod(actors-15, wonderfully-13)
amod(actors-15, talented-14)
nsubjpass(thrown-17, actors-15)
auxpass(thrown-17, are-16)
advcl(given-8, thrown-17)
nmod(thrown-17, into-18)
det(movie-20, a-19)
dep(into-18, movie-20)
cc(given-8, and-22)
nsubj(like-25, we-23)
nsubj(see-27, we-23)
aux(like-25, 'd-24)
conj:and(given-8, like-25)
mark(see-27, to-26)
xcomp(like-25, see-27)
mark(dominate-31, if-28)
nsubj(dominate-31, one-29)
aux(dominate-31, will-30)
advcl(see-27, dominate-31)
det(film-33, the-32)
dobj(dominate-31, film-33)
dep(see-27, both-35)
conj(see-27, provide-36)
det(entertainment-40, some-37)
advmod(good-39, pretty-38)
amod(entertainment-40, good-39)
dobj(provide-36, entertainment-40)

For every words dependency pair, we can simply add them into the collection of traditional bigram words-bag. Or we can also just build our words-bag collection only with words-dependencies data. We implement both methods and compare their precision in our experiment.

C. POS Tags of Words or Subjective vs. Objective

POS has long been a useful feature for many sentiment classifiers. Normally, POS can be used to remove ambiguity for words with different meanings.

Objective and subjective is a big problem for sentiment classification. Almost all the sentiment classifier focus on subjective text, because objective text normally has no relationship with author's emotions – they are just talking about the plot of the movie, or about another story. Thus, taking the objective text into the judgment of sentiment can be inaccurate. However, distinguishing between subjective and objective can be even harder than sentiment classification.

To some extent we can use POS tags to reduce the impact of subjective text. According to Pak and Paroubek [5], there exists a relationship between the POS of words and its feature

as subjective or objective. Subjective texts contain more personal pronouns, verbs in first person and especially, verbs in base form along with modal verbs. Objective texts contain more common and proper nouns and verbs in third person. We go through all the text in our dataset to calculate the algebraic relationship between every word and their polarity for subjective and objective. Thus we simply generate 5 patterns:

$$Subjective\begin{cases} personal\ pronouns \\ verbs\ in\ first\ person \\ verbs\ in\ base\ form\ with\ modal\ verbs \end{cases}$$

$$Objective\begin{cases} common\ and\ proper\ nouns \\ verbs\ in\ third\ person \end{cases}$$

Then we go through the texts to exam every word to see if it is in the five patterns. If the word belongs to the structure of subjective patterns, the word's value will be set to 1, while if the word belongs to the structure of objective patterns, its value will be set to -1.

V. CLASSIFIER

We use two methods to build classifier for the features extracted: SVM and Naïve Bayes. For SVM it's simple to make use of the features generated in section 4.1 to calculate. As for the Naïve Bayes:

$$P(c|d) = \frac{P(c)*P(d|c)}{P(d)} \quad (1)$$

It seems hard to directly apply our feature vector into Bayes rules. We can simplify our classifier by comparing $P(c_{pos}|d)$ and $P(c_{neg}|d)$. In such situation, $P(d)$ will make no difference for the result. Again we notice that our datasets contains same amount of positive reviews and negative reviews, so $P(c)$ also can be ignored. Thus:

$$P(c|d) \sim P(d|c) \quad (2)$$

To calculate $P(d|c)$, we go through all the features vectors in train set and get $P(f_i|c)$ for each feature f_i. Thus we get the final simplification of Naïve Bayes in our experiment as in (3):

$$P(c|d) \sim \prod_{i=1}^{n} P(f_i|c) \quad (3)$$

VI. EXPERIMENT & RESULT

A. Data and Preprocess

We use the movie review data set published by Pang and Lee [1] to conduct our experiment. The dataset contains 1000 positive and 1000 negative reviews about movies.

In order to make our classifier more accurate, we conduct some preprocess on the dataset.

1) We remove some unrecognized words, and words that appear no more than once.

2) We extend those short forms. For example, "don't" will be transformed into "do not". Thus words can be more comparable.

3) We correct the words which were apparently spelled wrong.

Then, we execute syntax parse on every data and generate syntax trees for every file. One file contains one movie review. We separate the data into train sets and test sets. The train sets take 80%, namely 1600, while the test sets take 20%, namely 400. With all these preprocesses, we now can apply our machine learning methods on theses files.

B. Experiment

We evaluate the result by calculating the accuracy on test set (200 files). There are two aspects we need to change in order to generate different test cases:

1) Words-bag: We apply three methods to generate words-bag collection – unigrams, bigrams and bigrams plus words dependencies. For each aspect we also combine clause tags to generate words-bag. Such tags mean when a word is in the main clause, it will be added a "-m". Meanwhile, if it is in a subordinate clause, it will be added a "-s". So we can generate different words-bag for the same word according to its position in the sentence.

2) Features: Again, we employ two methods to generate it – normal, normal + POS. In such situation, normal method means generating features according to the presence of words. We use presence instead of frequency because Pang [1] has proven that presence performs better than frequency. POS means variable values for each word according to its POS. Thus by using POS feature we can distinguish different POS of words in the words' feature vectors. The results are shown in Table II.

C. Result

1) Unigrams vs. Bigrams:

In the TABLE II, comparing row (1) with (4), (2) with (5), (3) with (6), we can see that firstly plain bigrams work poorer than plain unigrams. However, after we apply clause tags into words-bag collection and POS features into feature vector, bigrams method works almost the same with unigrams method.

2) Bigrams vs. Bigrams + Words Dependencies:

Comparing row (4) with (7), (5) with (8) and (6) with (9), we can observe an obvious variety between the accuracy of bigrams method and that of bigrams + words dependencies method. It is believed that the words dependencies really make the logistic relationship between words function in the sentiment classification.

3) Clause Tags and POS:

As can be seen from contraction in data from TABLE II, clause tags do not show much value as for the improvement of accuracies of algorithm. However, POS help the classifier improve much in its accuracy.

4) NB vs. SVM:

TABLE II. Precision of sentiment classifier on different words-bag, different features and different machine learning methods.

	Words-bag	*Features*	*NB*	*SVM*
(1)	unigrams	normal	80.0%	81.5%
(2)	unigrams	normal + POS	82.0%	84.5%
(3)	unigrams + clause tags	normal + POS	80.0%	85.5%
(4)	bigrams	normal	77.5%	79.0%
(5)	bigrams	normal + POS	82.5%	84.0%
(6)	bigrams + clause tags	normal + POS	81.0%	84.0%
(7)	bigrams + words dependencies	normal	81.0%	83.5%
(8)	bigrams + words dependencies	normal + POS	84.0%	85.5%
(9)	bigrams + words dependencies + clause tags	normal + POS	85.5%	86.0%

In the overall results, Naïve Bayes can't beat SVM in our experiment. This could result from the fact that both the bigrams and unigrams are not conditionally independent, which could violate the conditional independence assumption of Naïve Bayes.

VII. CONCLUSION

Words-bag method with machine learning techniques has long been widely used for sentiment classification. However, most classifiers have not taken syntactic features of text into consideration. They rely more on POS and other statistics features. We introduce syntactic features into our classifier, along with POS tags.

We use the dataset published by Pang and Lee [1]. The dataset contains 2000 movie reviews, half of which are positive while the other half are negative. Preprocess are conducted firstly to remove unrecognized words in English, extend short form words, and correct wrongly spelled words.

Syntax trees are constructed firstly and words dependencies are also generated to reveal the grammatical and logistic relationship between words in sentences. Then features about clause information as well as POS tags can be generated, and words-bag collection can be optimized by using bigrams with words dependencies. Then SVM and Naïve Bayes are applied in our experiment. From the result we observe that words dependencies and POS tags does improve the accuracy of bigram method. As for the clause features, we don't see much optimization.

REFERENCES

[1] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques." Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. Association for Computational Linguistics, 2002.

[2] Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." Proceedings of the conference on empirical methods in natural language processing (EMNLP). Vol. 1631. 2013.

[3] Dave, Kushal, Steve Lawrence, and David M. Pennock. "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews." Proceedings of the 12th international conference on World Wide Web. ACM, 2003.

[4] Mullen, Tony, and Nigel Collier. "Sentiment Analysis using Support Vector Machines with Diverse Information Sources." EMNLP. Vol. 4. 2004.

[5] Pak, Alexander, and Patrick Paroubek. "Twitter as a Corpus for Sentiment Analysis and Opinion Mining." LREC. Vol. 10. 2010.

[6] Ratnaparkhi, Adwait. "Learning to parse natural language with maximum entropy models." Machine learning 34.1-3 (1999): 151-175.

[7] Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann. "Recognizing contextual polarity in phrase-level sentiment analysis." Proceedings of the conference on human language technology and empirical methods in natural language processing. Association for Computational Linguistics, 2005.

[8] Zhan, Wei, Peifeng Li, and Qiaoming Zhu. "Sentiment classification based on syntax tree pruning and tree kernel." Web Information Systems and Applications Conference (WISA), 2010 7th. IEEE, 2010.

[9] Nakagawa, Tetsuji, Kentaro Inui, and Sadao Kurohashi. "Dependency tree-based sentiment classification using CRFs with hidden variables." Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2010.

[10] Duric, Adnan, and Fei Song. "Feature selection for sentiment analysis based on content and syntax models." Decision Support Systems 53.4 (2012): 704-711.

[11] Aue, Anthony, and Michael Gamon. "Customizing sentiment classifiers to new domains: A case study." Proceedings of recent advances in natural language processing (RANLP). Vol. 1. No. 3.1. 2005.

[12] Yang, Hui, Jamie Callan, and Luo Si. "Knowledge Transfer and Opinion Detection in the TREC 2006 Blog Track." TREC. 2006.

[13] Dunning, Ted. "Accurate methods for the statistics of surprise and coincidence." Computational linguistics 19.1 (1993): 61-74.

[14] Whitelaw, Casey, Navendu Garg, and Shlomo Argamon. "Using appraisal groups for sentiment analysis." Proceedings of the 14th ACM international conference on Information and knowledge management. ACM, 2005.

[15] Prabowo, Rudy, and Mike Thelwall. "Sentiment analysis: A combined approach." Journal of Informetrics 3.2 (2009): 143-157.

[16] Turney, Peter, and Michael L. Littman. "Unsupervised learning of semantic orientation from a hundred-billion-word corpus." (2002).