CS 224N Final Project
Boost up! Sentiment Categorization with Machine Learning Techniques

Andrés Cassinelli, Chih-Wei Chen

June 5, 2009

Abstract

We address the problem of categorizing documents by overall sentiment into two classes (i.e. positive or negative) and into multiple classes (e.g. one to five stars). We apply machine learning techniques to categorize a data set of movie reviews. In particular, we use a boosting algorithm. For determining the polarity of a review, we found that the algorithm has an interpretation similar to previous work in sentiment analysis, yet it achieves better accuracy in a more efficient way. Similar results can be seen when we apply the techniques to the multi-class categorization task. Without explicitly using the relationships of different labels during training, our classifier can discover the sentiment affinity between categories.

1 Introduction

Given the vast amount of unstructured information available online today, there is much to be gained from the development of automated systems that can effectively organize and classify this data, so that it can be leveraged by human users in a meaningful way. While it can be useful to categorize this kind of information according to its subject matter, classifying it according to the authors' opinions, or sentiment, can also provide researchers, business leaders, and policy-makers with valuable information ranging from rates of customer satisfaction to public opinion trends. Sentiment analysis has drawn great interest in recent years because of the surge of subjective content (blog posts, movie and restaurant reviews, etc.) being created and shared by Internet users, and the scope of new applications enabled by understanding the sentiments embedded in that content. For example, extracting the sentiment of a review can help provide succinct summaries to readers, and can be very useful in automatically generating recommendations for users.

Sentiment classification can also help determine the perspective of different sources of information, and yet another possible application would be the processing of answers to opinion questions. Specifically within the field of reviews, the numerical ratings that come

with many of them enable us to categorize them into finer-grained scales than just positive or negative categories. This richer information makes it possible to rank items or quantitatively compare opinions of several reviewers, thus allowing more nuanced analyses to be carried out.

2 Background and Related Work

While movie reviews may not be the domain where the application of sentiment analysis has the greatest potential to generate value, monetary or otherwise, it is an interesting and challenging testing ground for different sentiment classification approaches. After evaluating the performance of a simple unsupervised learning algorithm in the binary sentiment analysis of reviews across four different domains, Turney (2002) found movie reviews to be the most difficult. It was hypothesized that this was due to the tendency of reviewers to rate the individual elements of a movie differently from the movie as a whole within the same review, and addressing this issue remained a matter of future work. However, as Pang et al. (2002) suggest, the machine learning methods and features used when classifying movie reviews do not have to be specific to that domain. If a significant level of accuracy can be achieved in this difficult domain through a generalizable approach, then the benefits may easily transfer to other areas where sentiment classification can be applied.

For binary sentiment classification, Turney proposed counting the positive and negative terms and expressions in a review to determine its polarity. This idea was then augmented by Kennedy and Inkpen (2006), who also took contextual valence shifters, such as negation words, intensifiers, and diminishers, into account, and managed to improve the accuracy of the system. In this case, positive and negative terms were identified by querying a collection of dictionaries. In the same paper, a second method that uses Support Vector Machines (SVMs) was also proposed, and it was shown that this machine learning algorithm performs significantly better than the term-counting method with valence shifters.

As far as machine learning methods for sentiment analysis are concerned, Pang et al. (2002) and Pang and Lee (2004) have compared the performance of various classifiers when determining the sentiment of a document, and also found that SVMs were generally the best approach. Unigrams, bigrams, part-of-speech (POS) tags and term positions were considered as features, and using unigrams alone yielded the best results. Thus, SVMs have repeatedly emerged as remarkably effective tools for performing sentiment analysis, even when using very simple features in the classifier.

3 Problem Statement and Data Set

We propose to further improve the aforementioned systems by applying machine learning techniques to learn the weights for each term, since these terms may not be equal in their strengths. For the binary classification task (i.e. determining whether reviews are positive or negative), we will use a data set of classified movie reviews prepared by Pang and Lee (2004).

The data set contains 1,000 positive and 1,000 negative reviews collected from Internet Movie Database (IMDb) archive `rec.arts.movies.reviews`.

For multi-class sentiment analysis, Pang and Lee (2005) provided another data set with normalized subjective ratings on a numerical rating scale. They applied a meta-algorithm based on a metric labeling formulation of the problem. The ratings were quantized to three classes and four classes, respectively, for different experiments.

4 Method

Our aim is to solve the sentiment classification task for movie reviews using machine learning techniques. Under a machine learning framework, we have a dataset of m instances $((x_1, y_1), \ldots, (x_m, y_m))$, where x_i is the feature vector extracted from the ith data instance, and y_i is the label for that particular data instance. The feature can be, for example, a boolean value indicating the presence of the unigram word "great"; the number of appearances of the bigram "fully awesome" in the document; or any other convoluted value that captures the characteristics of the data instance. We will explain how features are selected for our work. The labels in our task are "positive" or "negative" for binary classification, and a number of stars for estimating the numerical rating of a movie review on a given scale.

4.1 AdaBoost

The term-counting method counts the positive and negative words in a review, and the polarity of the review is determined by the label with the most occurrences. However, not all positive and negative words have the same strength. For example, both "good" and "excellent" are both regarded as words with a positive sense, but the latter, when found in a movie review, is more likely to lead to a positive overall review than the former. Similarly, the negative word "insipid" does not reflect a reviewer's dislike towards a movie as strongly as the word "awful." Moreover, the term-counting method requires us to determine the polarity of each word in advance, yet the semantic polarity of a word depends highly on its context. For example, the word "predictable" might be neutral or even positive in general, but it usually has negative meaning in movie reviews, as in "the plot is too predictable." What makes the method even less ideal is the size of the candidate vocabulary. The classification time is linear in the size of our vocabulary, so it is desirable to keep the number of words that we have to consider small.

We propose to learn not only polarity, but also the weights of the words by applying machine learning techniques. Moreover, we want the learning algorithm to select a small set of discriminative words. From a classifier's point of view, each positive or negative word serves as a weak classifier. The appearance of each individual word in a movie review probably approximately indicates the polarity of the review. Boosting is a meta-learning method that tries to build a "good" learning algorithm based on a group of "weak" classifiers. It has been explored widely by many researchers. The most popular one among all boosting algorithms is AdaBoost, which was introduced by Freund and Schapire (1995). We will

use one of its many variants, AdaBoost.M1. The algorithm of AdaBoost.M1 is shown in Algorithm 1.

Algorithm 1 AdaBoost.M1 Algorithm

Input training set of m examples $((x_1, y_1), \ldots, (x_m, y_m))$
Initialize $D_1(i) = 1/m$ for all i.
for $t = 1,\ldots,T$ **do**
  Call weak learner, providing it with the distribution D_t.
  Get back a hypothesis $h_t : X \to Y$.
  Calculate the error of $h_t : \epsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$.
  if $\epsilon_t > 1/2$ **then**
    Set $T = t - 1$ and abort loop.
  end if
  Update distribution $D_t : D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$,where Z_t is the normalization constant.
end for
Output the final hypothesis: $h_{fin} = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}$

The AdaBoost.M1 algorithm works by calling a weak classifier several times. For our task, we simply let the weak classifier be an indicator function of a certain word. Each time a weak classifier is called, it is provided with a different distribution over the training examples. The idea of boosting is that it assigns higher probability to the part that it doesn't classify correctly, in the hope that the new weak classifier can reduce the classification error by focusing on it. The algorithm selects the weak classifiers to use, and learns the weight for each weak classifier. In the end, hypotheses from each iteration are combined into one final hypothesis.

Now, coming back to the perspective of term-counting method, the result of the AdaBoost.M1 algorithm has a beautiful interpretation. The AdaBoost.M1 algorithm returns a set of weak classifiers, each of them being an indicator function of a certain word, and the corresponding weight for each hypothesis. To determine the polarity of a review, we simply detect the presence of each word in the review, and sum up the weights. That is, we do a weighted counting:

$$\sum_{i=1}^{T} \beta_i \mathbf{1}(w_i) \tag{1}$$

where β_i is the weight for the ith weak classifier, and w_i is the word for that weak classifier. The indicator function returns 1 if the argument word exists in the document, and returns -1 otherwise. In general, the boosted classifier performs well even with only 10 iterations.

Therefore, we only need to count 10 weighted words in a document when determining its polarity.

Boosting has been very successful for solving two-class classification problems. To achieve multi-class classification, most algorithms have to restrict themselves to reducing the multi-class categorization problem to multiple binary classification problems. For AdaBoost, if the underlying weak learner is designed for multi-class classification (e.g., a decision tree), we can directly use the same AdaBoost algorithm for multi-class classification.

4.2 Feature Selection

For a machine learning algorithm to perform well, it is essential to have features that are descriptive of the data. In the framework of sentiment analysis, we need to be able to identify the words that express sentiment. Kennedy and Inkpen (2006) used the General Inquirer (GI) as their main source for identifying positive and negative terms. The GI system lists terms and their different senses. For each sense a short definition is provided, together with other information about the term. The terms are labeled as positive, negative, negation, overstatement, or understatement. Once a vocabulary of n words is obtained, for each document we can extract a n-dimensional feature vector, where the i-th index is 1 if the i-th word exisits in the document, and -1 otherwise. Instead of the presence of each word, we can also use the count of occurrences as the value at each index.

While the vocabulary obtained from GI may contain a rich set of words with sentiment sense, many of those words do not exist in our corpus. Therefore, we propose to form a set of words consisting of a predetermined number of the most frequent words in our corpus. A potential problem with using the most frequent words is that they might appear equiprobably in positve reviews and in negative reviews. Although the boosting algorithm promises to obtain a good classifier even if each weak classifier doesn't perform well, we still want to extract good features that can be used for other machine learning techniques. Fortunately, the boosting classifier allows us to pick the words efficiently. We simply start with all unigram indicators as weak classifiers, and let the boosting classifier automatically select the most descriptive ones.

4.3 SVM

Support Vector Machines are highly effective in many research and application domains, including text categorization. As mentioned earlier, they have been shown to outperform Naive Bayes and maximum entropy classifiers. The basic idea behind SVMs is to find a separating hyperplane with the largest margin in a given higher-dimensional feature space. The search for this hyperplane corresponds to a constrained optimization problem. Below, we compare the performance of the boosting classifier and SVMs, and we also show that we can incorporate the results of the boosting classifier with the SVM to achieve better performance.

5 Evaluation

We use the documents from the movie review corpus described in previous section. We test different unigram features on the binary classification task, and we also explore two different machine learning algorithms. We then try the features learned from binary classification on the multi-class categorization task, using a boosting algorithm.

5.1 Features

We use three sets of vocabulary. The first set consists of the positive and negative words from the General Inquirer. After removing duplicates (each of the meanings of words with multiple meanings had a separate entry in the original General Inquirer lists) and words which were featured in the lists, but not under their most common meaning, we were left with 1,565 positive terms and 1,896 negative words. A snippet of the vocabulary is shown in Table 1.

Polarity	Words
Positive	warm, unique, uncommon, talent, supreme, subtle, safe, plain, novelty, love
Negative	yawn, worse, wild, vomit, violent, unfair, superficial, sorrow, ridiculous, revenge

Table 1: Examples of Set 1: words from General Inquirer.

The second set of words includes the 3,102 most frequent words in the whole corpus, excluding words with length less than three. We list the top 10 most frequent words with length greater than three in Table 2.

The third set consists of 267 words learned from applying a boosting classifier to every word in the corpus. These words are selected automatically by the boosting classifier because they help distinguish the positive reviews from the negative ones. We've excluded named entities such as *Titanic*, *Spielberg*, and *Godzilla*. While these names of movie titles, directors or cast members might serve as good indicators for determining the polarity of a review (since not many people would consider *Godzilla* to be a good film, for example), we ignore them because we want this vocabulary to generalize well beyond the context of movie reviews. Therefore, we trim down the most discriminative vocabulary learned from the boosting classifier to mostly adjectives, adverbs, and verbs by removing named entities. We list some of the words in this set in Table 3. We will refer to the third set as *most discriminative words* hereafter.

5.2 Binary Classification with AdaBoost

We used the implementation of AdaBoost.M1 algorithm in the Weka suite of machine learning software (2005). For each experiment, we used three-fold stratified cross-validation to

Word	Appearance
that	15,105
with	10,778
this	9,560
film	8,849
movie	6,609
from	4,948
have	4,897
they	4,276
like	3,543
about	3,518

Table 2: Examples of Set 2: most frequent words with length greater than three in our corpus.

Polarity	Words
Positive	outstanding, avoids, astounding, fascination, finest, captures, supreme, wonderful, magnificent, uplifting
Negative	inexplicable, guilty, gags, catch, tired, pathetic, nowhere, predictable, illness, loser

Table 3: Examples of Set 3: most discriminative words learned from boosting algorithm.

evaluate the performance of the boosting classifer. The average results are reported.

We first used the GI word set, and varied the number of boosting iterations. The results are shown in Table 4. As we increase the number of iterations, the accuracy of the classifier goes up, since more and more weak classifiers are used to resolve the classification task. Boosting classifiers are well-known for their resilience against overfitting. It is important to note that using only words from GI, the classifier outperforms the term-counting based classifiers by Kennedy and Inkpen (2006), which used a larger vocabulary and takes into consideration valance shifters.

We listed the weak classifiers and their weights in Table 5. The first word picked by the boosting algorithm is *bad*, showing that, as expected, the appearance of this word is a good indicator of the polarity of the review. The second and third words that were picked were *worst* and *stupid*. Thus, if the reviewer says a movie is *bad*, but not the *worst*, then chances are it might still get a positive overall verdict.

As we increased the number of iterations, we also found that many weak classifiers are used repeatedly. This suggested that not many words can be used to differentiate between the two classes. Indeed, as we explained in the previous section, many of the words do not exist in our corpus, making it difficult for the classifier to use any of them to distinguish

Iterations	Class	Accuracy	Precision	Recall	F-score
10	Positive	0.657	0.637	0.728	0.679
	Negative		0.683	0.585	0.630
20	Positive	0.688	0.672	0.734	0.701
	Negative		0.707	0.641	0.672
30	Positive	0.714	0.696	0.758	0.726
	Negative		0.734	0.669	0.700
40	Positive	0.716	0.683	0.804	0.739
	Negative		0.762	0.627	0.688
50	Positive	0.728	0.717	0.752	0.734
	Negative		0.739	0.703	0.721
60	Positive	0.735	0.722	0.764	0.742
	Negative		0.749	0.706	0.727
70	Positive	0.745	0.730	0.777	0.753
	Negative		0.761	0.712	0.736
80	Positive	0.745	0.731	0.775	0.752
	Negative		0.761	0.715	0.737
Best Accuracy of	Positive	0.678	0.673	0.701	0.687
Kennedy and Inkpen (2006)	Negative		0.691	0.655	0.673

Table 4: Results for AdaBoost with different iterations, using the presence of words in GI set as features.

between positive and negative reviews.

This discovery motivated us to use words that are common in our corpus. In addition, we also used the words selected by the boosting algorithm as described in the beginning of this section. The results are shown in Figure 1. With only 10 iterations, the most discriminative set does not perform as well as the other two sets. As we run more iterations, however, it constantly performs better than the other two, except when we push the number of iterations to 80. At this point, the ability of the weak classifiers to help reduce the classification error is exhausted, and the most frequent word set becomes more effective, since it has the advantage of possessing a larger arsenal to tackle the job.

5.3 Binary Classification with SVM

For SVM, we used the SMO implementation in Weka. For each experiment, we used three-fold stratified cross-validation to evaluate the performance of the SVM classifiers. The results are shown in Table 6. Pang and Lee (2002) explored various features, including unigrams, bigrams, POS tags, etc., and three classification algorithms, including Naive Bayes, maximum entropy, and SVM, on the same data set. As previously mentioned, SVM performs best among the three algorithms in their experiments, and the best accuracy (82.9%) is

Word	Weight
bad	-0.51
worst	-0.53
stupid	-0.47
waste	-0.33
perfect	0.33
ridiculous	-0.23
awful	-0.25
memorable	0.25
outstanding	0.1
dull	-0.22
excellent	0.24
terrific	0.12
mess	-0.24
great	0.3

Table 5: The weak classifiers and their weights trained by AdaBoost.M1, using the presence of words in GI set as features.

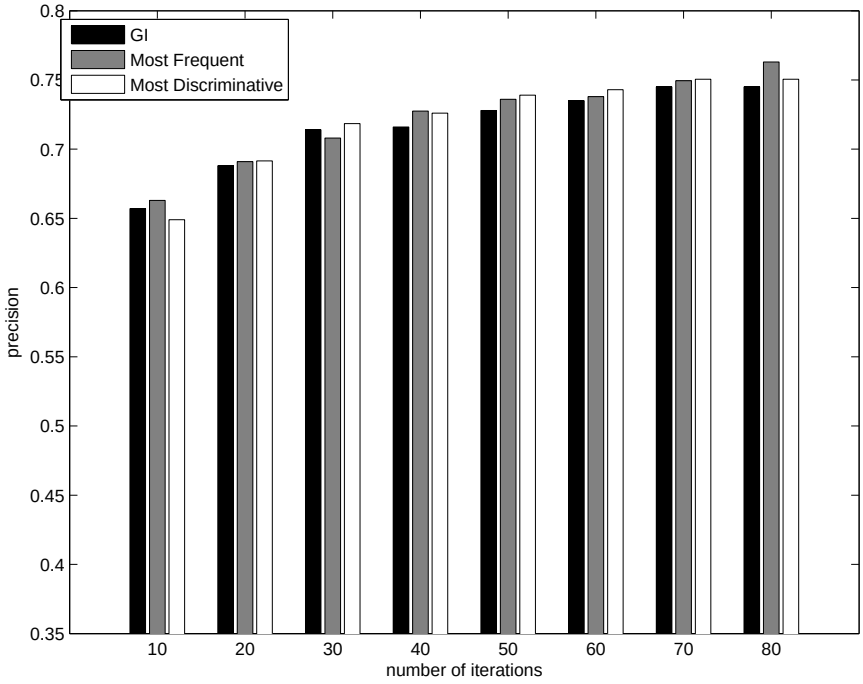

Figure 1: The precision of the classifiers using different sets of words as features.

achieved by using 16,165 dimensional feature vectors. Using the top unigrams achieves an accuracy of 81.4%. From Table 6, we observe that we are able to achieve high accuracy with low-dimensional feature vectors. The boosting classfier has identified a set of discriminative words that can also prove useful in other machine learning algorithms.

Finally, we investigated how boosting classifiers compare to SVMs. By increasing the number of iterations to 400, the boosting classifier performs almost as well as the optimal classifier, as shown in Table 6.

Method	Feature	Feature Dimension	Precision (%)
SVM	GI	3,461	73.8
SVM	Most Frequent	3,102	82.45
SVM	Most Discriminative	267	82.5
SVM	Most Frequent + Most Discriminative	3,369	83.55
Boosting	Most Frequent + Most Discriminative	3,369 (400)	82.95

Table 6: Results of SVMs with different features. We also compared the result with a boosting classifier trained with 400 iterations.

5.4 Multi-class Categorization with AdaBoost

We use the four corpora from four different authors described in previous section. In particular, we focus on categorizing the reviews into four classes: from 0 stars to 3 stars. In order to use a boosting classifier in multi-class task, we can no longer use our simple classifier. We replace the binary weak classifier with a multi-class decision tree. Specifically, we use the C4.5 (J48) classifier implemented in Weka. We set the number of iterations to be 10. After training, we would obtain 10 decision trees. Each decision tree determines the class of the document based on a series of questions about the presence of a certain word in that document. The final decision is determined by the weighted vote of all weak classifiers. We use the word set obtained from our binary classification work.

For each corpus, we ran ten-fold stratified cross-validations to evaluate the performance of the AdaBoost classifiers. The results are shown in Table 7.

Data Set	Precision
Dennis Schwartz	47.42 %
Steve Rhodes	45.20 %
James Berardinelli	51.03 %
Scott Renshaw	40.24 %

Table 7: Results of multi-class categorization on four corpora.

When training the classifiers, the four labels are given as if they were independent. We did not try to exploit the class relationships, although intuitively "three stars" should be closer to "two stars" than "zero star". Our multi-class classifier is capable of discovering the implicit relationship. We use the result from *James Berardinelli* to illustrate our discovery. The confusion matrix of the classifier is shown below.

0 star	1 star	2 stars	3 stars	← classified as
25	68	42	3	0 star
35	113	133	11	1 star
13	72	423	88	2 stars
3	14	158	106	3 stars

The classification output concentrates on the diagonal line of the confusion matrix. Thus, when documents are misclassified, they are more likely to be labeled as one of the classes adjacent to their true class. This suggests that our classifier captures the sentiment relationships among different classes.

6 Conclusions and Future Work

Our aim in this work was to apply generalizable machine learning techniques to improve on the results that had been previously obtained in the field of sentiment analysis in terms of accuracy, efficiency, or both. We succeeded in showing that a boosting classifier can achieve better accuracy than term-counting method, and that by learning the weights of different terms specific to a certain context, our classifier can adapt to classification tasks in different domains. The boosting classifier automatically learns a small set of discriminative words. Using this set of words, we were able to achieve an accuracy comparable to that of state-of-the-art SVM methods, while our approach has a better linguistic interpretation. Even though we did not achieve a higher accuracy than in the previous related work in the multi-class classification task, we were able to obtain results that were almost at the level of those in Pang and Lee (2005), but with a significantly less complex approach. Also, our classifier captures the sentiment relationships of different classes implicit in the labels given to the documents.

As Turney (2002) suggests, the system's performance could be improved by somehow tagging different sentences as referring to particular elements of the movie being reviewed, or to the movie as a whole. However, this would likely make the system significantly more complex, and how one might do this properly is still unclear, as Turney himself mentions. Moreover, introducing changes of this sort, which are based on relatively domain-specific linguistic insight, would make the approach less generalizable to other domains where sentiment classification could be of use. We therefore believe that it would be preferrable to approach future improvements by choosing even better features to use with machine learning algorithms, based not on characteristics particular to any one domain, but on common aspects shared across domains.

7 References

Alistair Kennedy and Diana Inkpen. 2006. *Sentiment Classification of Movie Reviews Using Contextual Valence Shifters.* In Computational Intelligence.

Yoav Freund and Robert E. Schapire. 1995. *A decision-theoretic generalization of on-line learning and an application to boosting.* In Computational Learning Theory: Second European Conference, EuroCOLT 95, pages 2337, Springer-Verlag.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. *Thumbs up? Sentiment Classification using Machine Learning Techniques.* In Proceedings of EMNLP.

Bo Pang and Lillian Lee. 2004. *A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts.* In Proceedings of ACL.

Bo Pang and Lillian Lee. 2005. *Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales.* In Proceedings of ACL.

Peter D. Turney. 2002. *Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews.* In Proceedings of ACL.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques, 2nd Edition.* Morgan Kaufmann, San Francisco.