Rotten Tomatoes: Sentiment Classification in Movie Reviews

Alyssa Liang
CS 229, December 15, 2006

1. Introduction

Text classification plays an large part in today's world, where the amount of information available is overwhelming. Although most text classification work is related to topical classification, the categorization of more subjective documents that depend more on style and the author's opinion is also important. Websites such as Amazon, IMDB, and Rotten Tomatoes rely on opinions and reviews to keep their sites running. Classification of documents by sentiment (i.e. positive or negative) could also be useful in recommender systems and customer service.

2. Previous Work

Most of the work in text classification has been topical classification. However, there has also been work in classifying text based on issues beyond subject matter, such as author identification and sentiment classification. Previous work in sentiment classification has mostly focused on classifying reviews as positive and negative. Pang, Lee, and Vaithyanathan have used learning algorithms such as Naive Bayes, SVM, and maximum entropy to classify reviews; however, their work focused on considering on the presence of terms (rather than their frequency) and using only a limited number of features. Pang and Lee have also gone further and worked on classifying reviews using a "multi-point" scale, such as the number of stars for a movie review. In addition, there also has been previous work (Turney, 2002) using unsupervised learning algorithms to classify reviews.

3. Movie Review Data

I'm using movie review data from http://www.cs.cornell.edu/People/pabo/movie-review-data/. The data consists of 1,000 positive reviews and 1,000 negative reviews. The data does not include reviews that could be considered "neutral." Movie reviews are an ideal source to experiment with sentiment classification: reviewers often have strong feelings about movies and even better, they usually provide a numerical rating along with the review.

One would first approach the problem by choosing words that would easily distinguish positive reviews from negative reviews, such as "bad," "awful," or "wonderful." The list of most discriminatory words (i.e., with the highest mutual information) bears this out. Interestingly, negative words dominate the list. From the reviews, it appears that negative reviews are more likely to be at an emotional extreme.

bad	perfect
worst	life
stupid	supposed
boring	memorable
ridiculous	dull
waste	poorly
awful	excellent
outstanding	perfectly
mess	script
lame	plot

Table 1: 20 words in the dataset with the highest mutual information

4. Machine Learning Algorithms

In this project, I'm focused on two learning algorithms: multinomial Naive Bayes classification and support vector machines. For these algorithms, I used the bag-of-words framework, although I also used word bigrams to

take context into account. Each feature represents a unigram (or bigram) and its value for a document is the number of times the feature appears in that document.

Naive Bayes

One of the simplest approaches to text classification is the Naive Bayes algorithm. It makes the assumption that the probability of one word appearing in a document doesn't affect the probability of another word appearing. Obviously, this isn't the case, but Naive Bayes still works well. For this project, I used the multinomial Naive Bayes classifier, which takes multiple occurrences of terms into account. Training the classifier is fairly quick and simple. It estimates the prior probabilities for a class c as

$$P(c) = \frac{N_c}{N}$$

where N is that total number of documents and N_c is the number of documents that belong to class c. The classifier also estimates the conditional probabilities that a term x appears in class c:

$$P(x \mid c) = \frac{n_{x,c} + 1}{n_c + |V|}$$

where $n_{x,c}$ is the total number of times term x appears in all the documents that belong to class c, n_c is the number of terms in all the documents that belong to class c, and $|V|$ is the size of the vocabulary. The conditional probabilities using Laplace smoothing to avoid zeros.

To classify a test document, we find

$$c = \arg\max_{c_j \in C} P(c_j) \prod_{i \in P} P(x_i \mid c_j)$$

where P is the set of all positions in the test document that contain a term in the vocabulary, and x_i is the term that occurs at position i.

In this project, I used the implementation of multinomial Naive Bayes from the libbow toolkit, as well as my own implementation.

Support Vector Machines

SVMs tend to work better than Naive Bayes. Unlike Naive Bayes, which is a probabilistic classifier, SVM attempts to find a hyperplane that divides the training documents with the largest margin. It finds the hyperplane using the SMO algorithm, and then it classifies a test document by finding

$$\sum_{i=1}^{m} \alpha_i y^{(i)} K(x^{(i)}, x) + b$$

where α_i and b are the parameters for the hyperplane. If the quantity is greater than zero, the the class y for the test document is set to 1; otherwise it is set to –1.

In this project, most of tests with the SVM classifiers used a linear kernel. However, I also ran SVM with a radial basis function kernel ($K(x,z) = \exp\left(-\gamma \|x - z\|^2\right)$ where gamma = 0.001); previous work (Siolas, d'Alche-Buc, 2000) has suggested that an rbf kernel works well. I used the SVM^{light} package to implement the SVM classifier.

5. Evaluation & Results

The data was split into 8 equal-sized folds (250 documents each), so that results are the average of the trials from eight-fold cross validation. I focused on using word unigrams and bigrams as the features. Unigrams + bigrams appears to have only a 1% or 2% improvement of unigrams only. The documents were parsed so that punctuation was removed, and unigrams that were less than 3 characters were removed from the vocabulary.

	Vocabulary Size	NB	SVM (linear)
unigrams	39,603	0.8280	0.7330
unigrams + bigrams	535,859	0.8395	0.7735

Table 2: Accuracies for Naive Bayes and SVM using unigram and unigram+bigram freatures.

Initial Unigram and Bigram Results

As expected, using both unigrams and bigrams improved the performance of the classifiers, since bigrams would take context of a word into account. Interestingly enough, though, Naive Bayes greatly outperformed SVM. Perhaps the multinomial model is better at modeling the data when using only simple unigrams and bigrams.

6. Improvements

	Naive Bayes	SVM (linear)	SVM (rbf)
unigrams + bigrams	0.8395	0.7735	0.6665
sentence upweighting (last 0.2)	0.8440	0.7915	—
sentence & adj/adv upweighting	0.8455	0.8090	—
tf-idf	0.8215	0.8460	0.8465
tf-idf + upweighting	0.8435	0.8690	0.8695
combined	—	0.8700	0.863

Table 3: Accuracies for learning algorithms with various improvements.

I attempted to improve the classification results using a number of different techniques, including upweighting and normalizing the counts.

Sentence Position Upweighting

Intuitively, the beginning and the ends of a movie review would be helpful in classifying it as positive or negative, since reviewers tend to summarize their views at those points. As a result, the classifiers might improve the performance if we put more weight on these sentences.

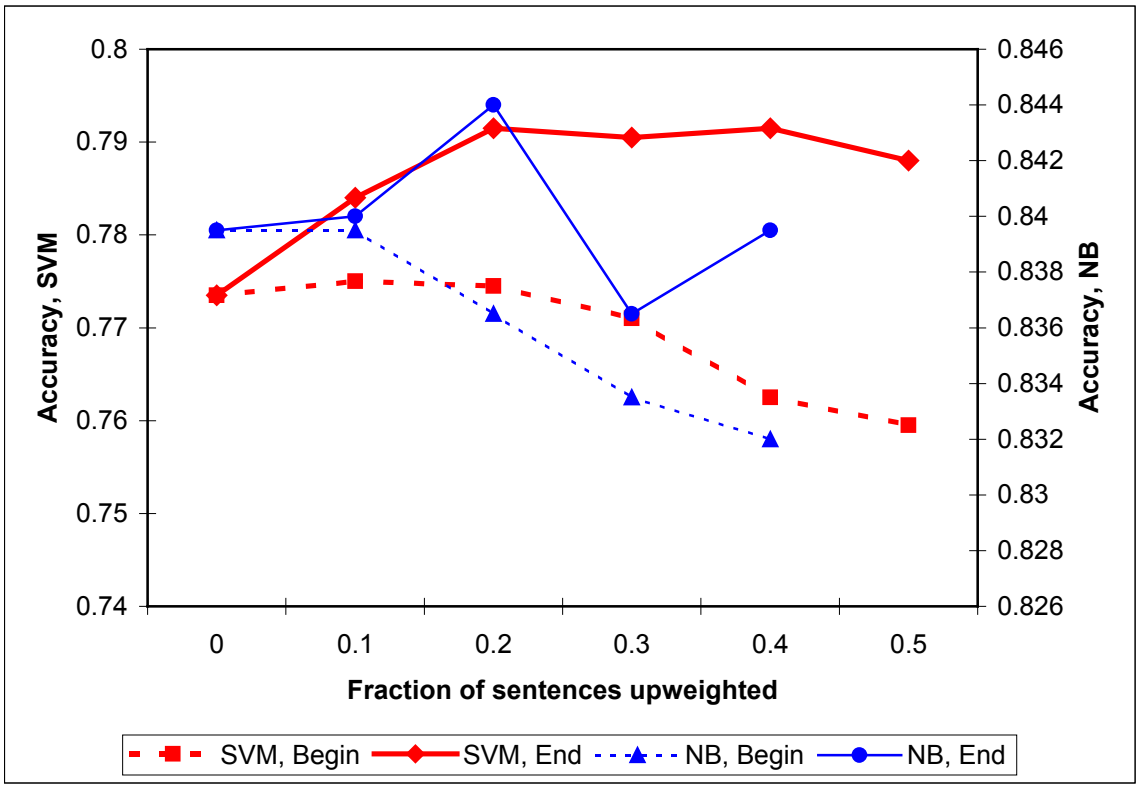

Figure 1: Accuracies based on fraction of sentences being upweighted.

Each term that appeared in one of the chosen upweighted sentences was counted twice, instead of only once. The graph above shows that when upweighting the ending sentences, both classifiers had the best results when they upweighted the 20% of the sentences. However, upweighting the beginning sentences actually decreased the accuracy. The movie review data showed that although the ends of the reviews did tend to summarize the author's opinion, the beginning the of the reviews usually summarized the movie itself.

The ideal proportion of sentences to upweight appears to be the last 20%. Implementing this upweighting improved both Naive Bayes and SVM. More sophisticated measures of position might help improve the accuracy even more, such as somehow incorporating a terms position into the term features. Ultimately though, the goal of the sentence upweighting is to identify parts of the review that give a good description of the reviewer's opinion. Identifying important parts of the reviews (perhaps by using summarizing techniques) might help improve the classification even more.

Part Of Speech Upweighting

The most discriminatory terms in the dataset were usually adjectives, such as "worst" or "outstanding." Adverbs, such as "poorly," also seemed helpful in determining how a reviewer felt about a movie. Therefore, I also applied upweighting based on whether a unigram was adjective or an adverb. I used the Stanford NLP Group Part-of-Speech tagger to get the part of speeach for the all the unigrams. If a unigram was an adjective or an adverb, I multiplied its count by 1.5.

POS upweighting improved the accuracy slightly; compared to the other improvements, it appears to be relatively useless. Intuitively though, using part of speech would seem to be a useful component to incorporate. Perhaps upweighting based on part of speech could be limited to only the most "useful" adjectives and adverbs.

TF-IDF

Instead of using the frequency counts of a term, I scaled it using its tf-idf weight (term frequency-inverse document frequency), which gives a better measure of how important the term is to the dataset. Term frequency was set as the number of times a term appeared in a document divided by the total number of terms in the document. Inverse document frequency measures how many documents a term appears in; an unimportant term, such as "the" or "and," would appear in many documents, and thus it would have a low idf.

$$\text{tf} = \frac{\#\text{ of occurrences of term in the document}}{\#\text{ of occurrences of all terms in the document}}$$

$$\text{idf} = \log \frac{\#\text{ of documents}}{\#\text{ of documents the term appears in}}$$

The new value for a term is set as $\text{tf} \times \text{idf}$. If we used only frequency counts, a term such as "the" would have a high value. However, tf-idf would scale the fequency count so it would have a much lower value.

For the SVM classifier, tf-idf weighting gave the single biggest improvement in accuracy, which points to its importance in improving accuracy. However, the accuracy of the Naive Bayes classifier actually decreased slightly. I speculate that using different equations for tf-idf would transform the data so that it would match the multinomial model better. For example, other work (Rennie, 2003) has suggested using

$$\text{tf} = \log(1 + \#\text{ of occurrences of term in the document})$$

Combining Learners

I hoped to combine the Naive Bayes and SVM classifiers to improve the results. I ran another SVM classifier over the outputs for the Naive Bayes and SVM classifiers. The inputs for the new SVM classifier were the two probabilities (for the positive and negative class) given by Naive Bayes and the output from the SVM with a linear kernel, and the SVM with a rbf kernel. The new SVM classifier used a linear kernel. It also used the results from the classifiers using tf-idf weighting and upweighting.

Combining the learners gave a very slight improvement; however, the improvement was so small that it only correctly classified two more documents compared to the original SVM classifier with a linear kernel.

7. Discussion

With the addition of the improvements, SVM outperformed Naive Bayes. Interestingly enough, the Naive Bayes accuracy increased only a little with the addition of the improvements, by less than 1%. In contrast, SVM showed huge improvements, by almost 10%. Without the improvements, SVM with the rbf kernel gave much worse results. However, once tf-idf weighting was implemented, it gave similar results to SVM with a linear kernel, and even slightly better.

The misclassified documents tended to be ones that were more ambivalent, and pointed out both positive and negative aspects of a film. In addition, misclassified documents would include the author's opinions on how contrasting viewers might react to the film; for example, an author might give a positive review, but say, "people who don't like gross-out humor will hate this movie." While it would be very easy for people to identify this sentence as irrelevant to the author's overall opinion of the movie, it seems as if it would be a very difficult task for a classifier. Sentiment classification could be improved by identifying the most relevant sentences in the review (such as sentences where the subject is the movie, which will probably be related to how the author feels about the movie).

References

Grilheres, B., Brunessaux S., Leray P. Combining Classifiers for Harmful Document Filtering. In RIAO 2004.
http://www.riao.org/sites/RIAO-2004/Proceedings-2004/papers/0131.pdf

Joachims, Thorsten. "SVM^{light}" 2004.
http://svmlight.joachims.org/

McCallum, Andrew Kachites. "Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering." 1996.
http://www.cs.cmu.edu/~mccallum/bow.

Pang, B., L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In EMNLP 2002, 79–86.
http://www.cs.cornell.edu/home/llee/papers/sentiment.pdf

Rennie, J., Shih, S., Teevan, J., Karger, D. Tacking the Poor Assumptions of Naive Bayes Text Classifiers. In ICML-2003.
http://haystack.lcs.mit.edu/papers/rennie.icml03.pdf

Siolas, G., d'Alche-Buc F. Support Vector Machines Based on a Semantic Kernel for Text Categorization. In IJCNN 2000.
http://ieeexplore.ieee.org/iel5/6927/18674/00861458.pdf

Toutanova, K. "The Stanford NLP Group Part-of-Speech Tagger." 2006.
http://nlp.stanford.edu/software/tagger.shtml

Turney, P. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In Proceedings of the 40th Annual meeting of the Association for Computational Linguistics (ACL), 07/2002. pg. 417-424.
http://oasys.umiacs.umd.edu/oasys/papers/turney.pdf