1 Introduction

Text classification has increasing potential applications in many aspects of information world, such as recommender systems and customer service. The goal of this assignment is to apply Naive Bayes classifier to a data set of labeled textual movie reviews and practice Scala/ScalaNLP. The data set “Polarity dataset v2.0” is from http://www.cs.cornell.edu/People/pabo/movie-reviewdata/, created by Bo Pang and Lillian Lee at Cornell. The reviews originally included numerical scores (−4 to +4), but they have been partitioned into positive and negative sets, and matched in size. Therefore the data contains 1000 positive and 1000 negative reviews all written before 2002.

2 Methods

2.1 Naive Bayes Algorithm

We apply standard Naive Bayes algorithm (multinomial model) to classify the movie review data set. In the training phase, conditional probabilities are estimated by counting the number of appearance of each word in positive and negative context respectively. These probabilities are further smoothed using a Laplace smoother, and are stored in scala map data structure. We score each document by adding up the log conditional probability of each word appearing in positive or negative context; a larger score indicates which sentiment context the document should be classified to.

The algorithm is optimized by blacking out stop words, such as “and”, “the” etc. A detail explanation of the algorithm is presented in the following pseudo codes.

Algorithm 1 Naive Bayes classifier training pseudo code

\[
l \leftarrow \text{LaplaceSmoother}
\]

for all \( c \in C \) do

\[
prior[c] \leftarrow 1/2
\]

\[
text_c \leftarrow \text{ConcatenateTextOfAllDocInClass}(c)
\]

for all \( w \in text_c \) do

if \( w \neq \text{stopword} \) then

\[
T_{cw} \leftarrow T_{cw} + 1
\]

end if

end for

\[
\text{condprob}[w][c] \leftarrow \frac{T_{cw}+l}{\sum_{w'}(T_{cw'}+l)}
\]

end for
Algorithm 2 Naive Bayes classifier applying pseudo code

\[
W \leftarrow \text{ExtractTokensFromDoc}
\]

\[
\text{for all } c \in C \text{ do}
\]

\[
\text{score}[c] \leftarrow \log \text{prior}[c]
\]

\[
\text{for all } w \in W \text{ do}
\]

\[
\text{score}[c] + = \log \text{condiprob}[w][c]
\]

\[
\text{end for}
\]

\[
\text{end for}
\]

\[
\text{return } \text{arg max}_{c \in C} \text{score}[c]
\]

2.2 F1 Measurement

Throughout the results part of this report, we use F1 measure to evaluate the performance of our algorithms. F1 measure is defined as follows.

\[
F = \frac{2PR}{P + R}
\]

where \( P \) and \( R \) are precision and recall respectively.

2.3 K-fold Cross-Validation

Cross-validation is a statistical method of evaluating and comparing the performance of learning algorithm(s) by dividing data into two segments: one used to learn or train a model and the other used to validate the model. In \( k \)-fold cross-validation the data is first partitioned into \( k \) nearly equally sized segments. Subsequently \( k \) iterations of training and validation are performed such that within each iteration a different fold of the data is held-out for validation while the remaining \( k - 1 \) folds are used for learning.

2.4 Stemming and Stop Words

In addition to standard Naive Bayes algorithm, we also apply stemming and filtering stop words to boost the performance of our classifier.

3 Results

3.1 Preprocessing of the data

Exploratory data analysis allows us to look at the variables contained in the data set before beginning any formal analysis. We first want to know if there is any difference between the two classes (positive, negative) in terms of number of sentences and words used. The reviews were tokenized and formatted as one sentence per line. Table 1 below indicates that there are no significant differences between the two classes on these features.

Then we proceeded to create a vocabulary from all of these documents. We tokenized each review, stemmed each token and removed the stop words (by applying http://www.textfixer.com/resources/common-english-words.txt). Eventually we obtained a vocabulary composed of 26048 stem terms, mapped with counts of each term and indices of the documents that contain the term.
### Table 1: Statistics for Review Documents

<table>
<thead>
<tr>
<th></th>
<th>Reviews</th>
<th>Sentences</th>
<th>Words</th>
<th>Average words per review</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>1000</td>
<td>32937</td>
<td>702431</td>
<td>702.43</td>
</tr>
<tr>
<td>Negative</td>
<td>1000</td>
<td>31783</td>
<td>629300</td>
<td>629.30</td>
</tr>
</tbody>
</table>

#### 3.2 Model Selection

We implemented Multinomial and Bernoulli model to train our Naive Bayes classifier. To evaluate the quality of our classifier, we did a 10-fold cross-validation. The data was split into 10 equal-sized folds (200 reviews each, including 100 positive and 100 negative reviews). In addition, we tried various alpha values for smoothing and evaluated performance of the system by F1 measurement. The figures below showed our final results of performance.

![Multinomial Model](image)

Figure 1: F1 measurements for multinomial model: The y-axis is F1 measurement averaged over ten-fold cross-validation results, the x-axis is value of alpha smoothing.
Based on the ten-fold cross validation estimated prediction accuracy by F1 measurement, we found that multinomial model performed relatively better than bernoulli model.

### 3.3 Feature Selection

Our vocabulary consists of totally 26048 word stems, which is a large feature set for classification. To find the best explanatory words that can predict the outcome, we first looked at the word stems with top weights (top conditional probabilities given either positive or negative class). However we find that both positive and negative classes share similar top terms.

**Top 15 Word Stems From Positive Reviews:**

```
"film" "one" "movi" "charact" "more" "out" "up"
"make" "time" "stori" "scene" "good" "see" "play"
"even" "well" "onli" "veri" "life" "much"
```

**Top 15 Word Stems From Negative Reviews:**

```
"film" "movi" "one" "out" "charact" "up" "make"
"more" "time" "even" "onli" "scene" "good" "play"
"bad" "look" "go" "much" "stori" "see"
```

(As we have performed stemmer on all the tokens, our vocabulary consists of word stems instead of entire words.)

After looking at these terms, we think it make sense that the terms with highest conditional probabilities were shared by both classes, since the conditional probability of a term depends on
the frequency of the term in a class. Therefore, the words that people tend to use more often when writing a typical style of document (i.e. review for movies) would outstand the rest of the vocabulary, regardless of the review polarity.

However, these common neutral words would be little help to distinguish class of a review, and too many of these overlapping terms with top weights will generate noise for accurate classification. Narrow down the range of feature term space towards may improve the classification accuracy. For example, one could choosing words that would easily distinguish negative reviews from positive reviews, such as 'worst', 'wasting,' or 'excellent'. A common feature selection method is to compute as the expected mutual information (MI) of term \( t \) and class \( c \). Because MI measures how much information a term contain about a class based on its distribution. Therefore a term with higher MI values making classification decisions. We calculated MI of each word stem in our vocabulary using the method in “Introduction to Information Retrieval, Chapter 13, Classification and Naive Bayes, C.D. Manning, P. Raghavan and H. Schutze”(refer to equation 13.17).

It appears that most of these words are adjectives likely to be at an emotional extreme. It suggest that we could put more weights on adjectives to improve system performance.

4 Discussion

Our experiments tend to classify movie reviews as either positive or negative. Multinomial model performed better than Bernoulli model.

Interested to know why some documents were misclassified, we read through these misclassified reviews. It seems most of these reviews contain more ambivalent opinions. The reviewers tended to cover both attracting and disappointing points of a movie. It is easier for human to find out the overall opinion of the author based on the context, but it would be very difficult for a classifier to identify.

To improve the performance of our classifiers, we can try unigrams plus bigrams rather than unigrams only as the features. We expect that including both unigrams and bigrams would improve the performance of our classifiers as bigrams take context of a word into account. For example, an expression such as "don’t like" or "would not hate" would be learned differently by classifiers using unigrams or bigrams. We may also improve the classification results by up-weighting terms with top MI values, since it appears that those emotional expressions are helpful in determining how an author felt about a movie. Furthermore, usually people tend to summarize at the beginning and/or ending of a review thus we can upweight the sentences in these two parts of a review.