CS 686 Data Mining Final Project Report

Xi Han
University of San Francisco

Luzhou Li
University of San Francisco

Chao Lu
University of San Francisco

ABSTRACT
People are commenting about movies on different kinds of social media. These comments could be clue on what people are actually thinking about this movie. Movie makers or Theaters would like to analyze this to decide whether they will make it a series or keep it showing for more days. Our task is to analyze the sentiments of movie comments in order to get a average score about this movie. This is an existing project here called <Sentiment Analysis of IMDB Reviews>[http://stanford.edu/~cpiech/cs221/homework/finalProject.html#ideas]

Author Keywords
Supervised learning; feature deduction; classifier

INTRODUCTION
We examine the reviews to observe what types of words/ sentences/phrases tend to indicate positive and negative sentiment. For example, the word "great" could indicate a positive sentiment towards the movie, unless it is part of the sentence "This movie was a great big disaster." What we are trying to do is develop a classifier from the training data which tries to predict the score of a review based on its text.

DATA
The original data comes from Stanford AI class. We are provided training dataset, test dataset, as well as word-score mapping.

For the training data folder, it has 12500 negative rated comments and 12500 positive rated comments. For the test data folder, it also has 12500 negative rated comments and 12500 positive rated comments. That is to say, we have 50000 labeled data in all.

Since we did our own part separately, the data interface we shared is csv file format instead of list. I understand that we can combine all 50000 comments into one large dataset, and do cross validation on it, but due to the argument we use (csv file name), it's hard to apply cross validation on our handmadeDecision Tree and Naive Bayes classifier. also considering the fact that given training data for training and given test data for testing. As a balanced result, we apply cross validation on the existing sklearn classifier.

DATA PREPROCESSING
As mentioned above, we have a list of word scores, and we have also 50000 labeled comment files. What we did first was to classify all the words into 10 different bags based on their own score value, each of them has over 1000 words, which means to preprocess the word scores into categorical label: 0, 1, …, 9 representing sentiment level from strong negative to strong positive, and then save to a dictionary for further use.

After this, we tokenize each comments into a list of words. We filtered the stop words and the punctuations simply using `regular expression tokenizer`. I admitted that the way of tokenizing is too simple to get a good classification score. Therefore, we can have more complex ways to tokenize the comments, as mentioned on class, adding negation filtering in the future.

By mapping these word categories dictionary, we generated a list of sentiment word counts, which has 10 columns representing how many words are under those 10 categorical labels. We use these data to test the Logistic Regression and SVM classifier.

In order to fit the data into our Decision Tree and Naive Bayes classifier, we change the previous data into binary one. It surely lost lots of information but it is the only way we figured out currently.

DECISION TREE
Decision tree is a kind of Supervised Learning. We choose decision tree as one of our model to train and test our movie reviews because our movie reviews are already turned to 0 and 1 values.

1. **Data**
The data which decision tree train and test has ten features. Each feature means a word package. There are two values of each word package, zero and one. One means that the word of this package exists in the movie review. Zero means that the word of the package does not exist in the movie review.

2. Information Gain

$$H(S) = -\sum_{i=1}^{n} P(u_i) \log P(u_i)$$

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

We calculate the information gain of every feature. Choose the max one as our first tree node. Then the value decides which node to go next.

3. Split by values
After we get the feature with max information gain, we split the data into two parts by the value zero and one. Calculate each information gain of these two parts. Then we get max information features based on these two parts, and these two feature are next layer of the root node.

4. Recursive process
We recursively do 2 and 3 steps, then a tree can be built. And the tree must look like a binary tree because the values are all binary.

5. ZeroR
At last, the information gain may be zero, which means the values are all ones or zeros. Then we need to use zeroR to check which value of label has a larger number. And the value is the prediction value of the leaf node.

6. Evaluation and Improvement
The accuracy is shown below:

```
accuracy:
0.75948
Press any key to continue . . .
```

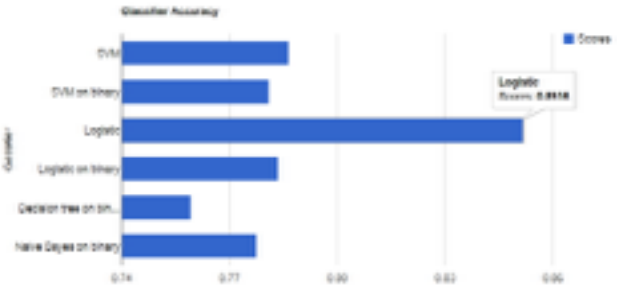
It seems not so good. I think the reason is that we use the word package which is lost some information. To improve this, I think we can use specific words in stead of the word packages. For example, use 200 negative words and 200 positive words as the features.

NAIVE BAYES
There are two different ways to fit this model: one is to generate a bag of words by ourselves, either by the highest positive or negative scores, or use term frequency and inverse document frequency to find out the most important words that can differentiate the attitude of a specific movie review. Another approach would be use binary feature, which use 0 or 1 to represent whether or not such review has the word from each bag of words. To make sure that different data models are provided equal change to win, we choose the second one, also know as the one used by decision tree. We do notice that in such way we will lose certain amount of information that could be quite influential to our final result, but from the experiments we can see that it's not a bad idea to execute the feature deduction, especially when the original dataset is large enough. The accuracy is shown as below:

```
/System/Library/Frameworks/Python.framework/Vers
0.777671106844

Process finished with exit code 0
```

COMPARISON
Other than decision tree, naive bayes, we also import the support vector machine and maximum entropy from sklearn, the last one is also known as logistic regression, whose basic idea is that we put our features into a math function, at every iteration we fine tune our parameter until the curve on the x-y coordinate system doesn't vibrate that much any more, then we set up a value differentiate between multiple labels. The next things is to put test data into that equation and see what label it should be. Among all of these classifier model, maximum entropy finally win the competition with a average accuracy higher than 85%. We are not certain it's because the model is better, or it's more suitable for the features we created. But one thing we can tell is that other models has a similar result in this case, which maximum entropy is in a obvious higher level.



REFERENCES
1.Das, Sanjiv R., and Mike Y. Chen. "Yahoo! for Amazon: Sentiment extraction from small talk on the web." *Management Science* 53.9 (2007): 1375-1388.