**Opinion Mining term project report**

Submitted by Ms. Pham Thi Mai Phuong 56070503447

Lecturer: Assistant ProfessorDr.Nuttanart Facundes

**Introduction**

In movie industry, for both film producers and audiences, movie reviews play an important role. On the producer's side, movie reviews help them adjust their marketing strategies. Moreover, movie reviews have a significant impact over the box revenue. On the audience's side, movie reviews help them to decide whether a movie is worth watching.

In the era of computer and the Internet, information and opinions on movies can be easily shared among people around the world. However, the abundance of information can become a problem since it confuses people. Additionally, sometimes it is not possible to do exhaustive survey.

To target this problem, this project will work on sentiment analysis on iMDb movie reviews. The target of this project is to create a small model that is able to classify movies sentiment reviews into the correct category (positive or negative).

Related work

Kang et al. (2012) [1] attempted to classify 70,000 restaurant reviews with star scores into positive and negative classes. These reviews were pre-labeled according to the associated star scores. The reviews with star score equal to 1 were handled as negative and reviews with the star scores of 5 were handled as positive. However, the positive reviews that had the positive and negative words appear approximately the same number of times were discarded from the experimental data. The preprocessing step eliminates the special symbols, HTML tags and other unnecessary characters that are included in each selected review. 10-fold cross validation method were used to split the experimental data into training and testing sets.

In feature extraction step, a set of unigrams, *bigram and unigrams+ bigram patterns* were extracted from each review the experimental data. Unigram pattern consists of one word, for example, good or delicious. Bigram pattern consists of two word, for example, very excellent or really good. Each pattern is made up of a set as follows.

L = {TYPE; TARGET; PATTEN; POLARITY}

Where TYPE indicates the pattern type, which is unigram, bigram or both; TARGET indicates the attribute of an entity, for example the food’s taste, facility, mood and price; PATTEN indicates the real value for each pattern, for example good or delicious and POLARITY indicates the label for this pattern, which is either positive or negative.

Data classification consists of a learning phrase and a classification phrase. In the learning phrase, the research attempted to build classification models with both *K-mean* and *improved* *NaiveBayes* and compare the results between the two using WEKA.

K-mean is tried to find an optimal hyperplane for clustering the input documents into the two classes. That is, the distance among documents in the same cluster would be minimized and the distance among documents in different clusters would be maximized.

NaiveBayes classified the input documents based on *the probability that a document belonged to each class*. The input belongs to the class with higher probability. *The probability that a document was positive and negative when it contained a particular pattern explored in the feature extraction* step was calculated. The probability that a document belonged to the positive and negative class was the multiplication of all resulting probabilities from the previous calculation.

Improved NaiveBayes was NaiveBayes, but the probability that a document was positive and negative was emphasized by the portions which positive patterns and negative patterns appear in the document.

The experiments in this research shown that the accuracy obtained from NaiveBayes was around 80% and was generally higher than that of K-means.

**Method**

**The dataset**

**Source**: Pang/Lee ACL 2004. Released June 2004 [3]

The dataset is a set of text files consists 2000 labeled examples extracted from the Internet Movie Database (IMDb) archive [2]. Each line in a text file corresponds to a single sentence, as shown in figure 1. Preliminary steps attempted to remove rating information from the text files. However, only the rating information upon which the rating decision was based is guaranteed to have been removed. Therefore, if the original review contained several instances of rating information, potentially given in different forms, those were not recognized as valid ratings remain part of the review document and thus would not be removed.

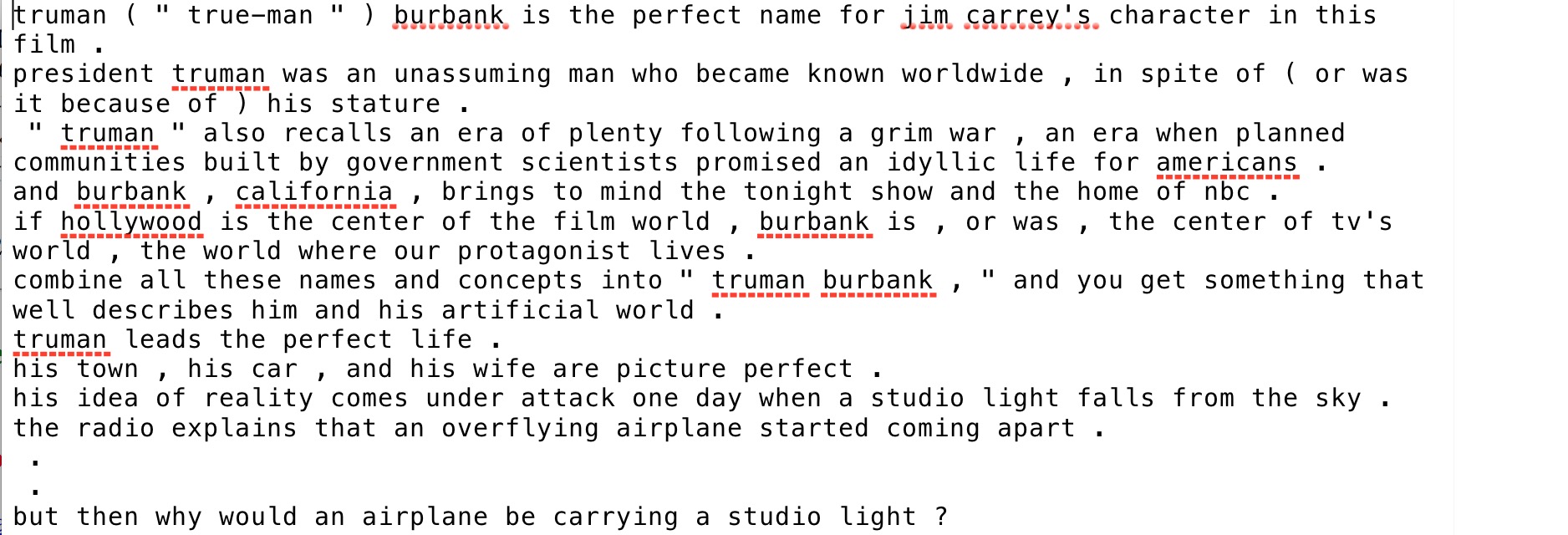


Figure 1 An example of a review document

The dataset is labeled into two classes: ‘pos’, stands for positive and ‘neg’, stands for negative. For storing convention, positive review documents are store in ‘pos’ folder and negative review documents are store in ‘neg’ folder.

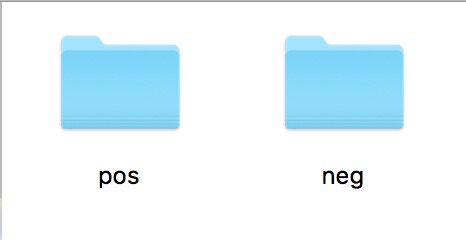


Figure 2 Label for positive class and negative class

**Rating Decision**

* With a five-star system (or compatible number systems): three-and-a-half stars and up are considered positive, two stars and below are considered negative.
* With a four-star system (or compatible number system): three stars and up are considered positive, one-and-a-half stars and below are considered negative.
* With a letter grade system: B or above is considered positive, C- or below is considered negative.

**Data pre-processing**

In order to make the data more robust for analysis, we will need to perform some preprocessing tasks. We will do this using WEKA.

The dataset file format is not supported by WEKA. In order to process this dataset with WEKA, we will need to convert the text files into a single arff file. Arff file format is used in WEKA because it is less memory intensive, faster and better for analysis. It contains a relation name, a review attribute, a class attribute and the data. Each tuple in this relation is a review from a text file.

We will also need to add some syntax in order to make the review documents WEKA-recognizable. The syntaxes are as follows. Strings need to start and end with a single quote. Apostrophes need to have slashes in front of them. And each document is marked with a positive or negative label.

WEKA provides a command to do this, as follow.

java weka.core.converters.TextDirectoryLoader /Users/phuongpham/Library/Mobile\ Documents/com\~apple\~CloudDocs/CPE/Opinion\ Mining/project/review\_polarity/txt\_sentoken **>** /Users/phuongpham/Library/Mobile\ Documents/com\~apple\~CloudDocs/CPE/Opinion\ Mining/project/iMDb.arff

This command imports the 2000 review documents into a single arff file and adds some syntaxes in order to make the arff file WEKA-recognizable.

Below is an example of arff data after the first step of pre-processing.

@relation \_Users\_phuongpham\_Downloads\_review\_polarity\_txt\_sentoken

@attribute text string

@attribute @@class@@ {neg,pos}

@data

'so ask yourself what \" 8mm \" ( \" eight millimeter \" ) is really all about . \nis it about a wholesome surveillance man who loses sight of his values after becoming enmeshed in the seedy , sleazy underworld of hardcore pornography ? \nis it about the business itself , how , bubbling just beneath the surface of big-town americana , there\'s a sordid world of sick and depraved people who won\'t necessarily stop short of murder in order to satisfy their sick and twisted desires ? \nor is it about those who can , those who are in a position to influence the making of the kinds of films sick and demented people want to see ? \ni\'m not talking about snuff films , supposed \" documentaries \" of victims being brutalized and killed on camera . \ni\'m talking about films like \" 8mm \" and its director , joel schumacher . \nwith a recent run of big budget movies to his credit-- \" batman & robin , \" \" a time to kill , \" \" batman forever , \" \" the client \" --schumacher certainly has that kind of influence . \nis \" 8mm \" something you really want to see ? \nprobably not . \nthe first two-thirds of \" 8mm \" unwind as a fairly conventional missing persons drama , albeit with a particularly unsavory core . \nthen , as it\'s been threatening all along , the film explodes into violence . \nand just when you think it\'s finally over , schumacher tags on a ridiculous self-righteous finale that drags the whole unpleasant experience down even further . \ntrust me . \nthere are better ways to waste two hours of your life . \nnicolas \' \" snake eyes \" \' cage plays private investigator tom welles who is hired by a wealthy philadelphia widow to determine whether a reel of film found in her late husband\'s safe documents a young girl\'s murder . \nwelles goes about his assignment rather matter-of-factly , and the pieces of the puzzle fall into place rather neatly , almost as if you don\'t need any specialized skills or training to do this . \nwelles certainly makes it look easy . \nand cops , obviously , never look in toilet tanks for clues . \nthe deeper welles digs into his investigation the more obsessed he becomes , like george c . scott in paul schrader\'s \" hardcore . \" \noccasionally , a little flickering sound whirs in his head like sprockets winding through a film projector , reminding him of his unpleasant task . \nthere are hints that this is taking its toll on his lovely wife , played by catherine keener , who is frustrated by her husband spending all of his time in cleveland rather than in their ugly split-level home in harrisburg , pa . \n \" 8mm \" doesn\'t condemn or condone its subject matter , it just exploits it . \nthe irony , of course , is that schumacher and \" seven \" scribe andrew kevin walker\'s vision of life in the snuff lane is limited by what they can show in an r-rated , first-run hollywood product . \nso we only see snippets of snuff , and a lot more footage of nicolas cage covering his face in horror . \nlater it\'s the turn of joaquin phoenix ( who\'s quite good and by far the film\'s most interesting character as adult bookstore flunky max california ) to cover his face as the horrid thing is screened over and over again . \nall this to get to the familiar yet offensive \" revelation \" that sexual deviants are not , indeed , monsters but everyday people like you and me . \nneither super nor standard, \" 8mm \" is shocking only in its banality . \n',neg

Nevertheless, it is difficult to build a model if we left the text in each movie review remained as a mere string. In the next step, we choose some important features in the dataset and let the values each document has for these features represent each review document. We will refer to the features as **attributes** in accordance to the terminologies in data mining. The right number of attribute has to be specified so that the model is not fed with too many parameters, which can lead to overfitting. The decision on which attribute to keep could base on the followings.

1. **Ignore irrelevant words**

Most words, such as “as”,”is”, can be irrelevant to decision making tasks. Thus, each document needs to be filtered with a pre-initialized *“stop words” list* so that *only useful* *words are kept* for decision making.

An English *“stop words” list* is provided within WEKA. However, WEKA also allows us to import a different stop word list. For this project, we will use the stop word list provided by WEKA.

We will experiment the dataset both with and without using the “stop words” list to see whether the list has any significant effect on the accuracy.

1. **The number of times a word appears in a document**

A certain word has to appear more than an n time(s) in order to be consider as an attribute. We will experiment the dataset both with and without setting up the n value. At the same time, we will gradually increase the value of n to see if it is going to have any significant effect on the accuracy.

1. **The relationship among words and documents**

This is equivalent to the answer to the question ‘how often a term is in a document?’. The more times a word appears in a document, the more chance it is important to the document. However, we may also need to take into consideration the length of each document. A word appears three times in a 200 character-long document would have more effect this document than to a document with 2000 character-long. We will experiment the dataset both with and without testing on the relationship between a word and a document and see whether the relationship has any significant effect on the accuracy.

1. **The use of tokenizer**

Tokenizer is the method we use for splitting up the text into token. NGramTokenizer finds potentially predictive n-words unit and set them as attributes. Examples of NGramTokenizer are ‘al’, ‘by th’, ‘becaus’.

AlphabeticTokenizer finds potentially predictive single word unit as an attribute. Examples of AlphabeticTokenizer are ‘apple’, ’care’, ’boat’.

WordTokenizer finds potentially predictive single word unit as an attribute. However, it is different from AlphabeticTokenizer in that it will create words that contains characters does not belong to the alphabet, such as ampersand, at sign or hyphen. AlphabeticTokenizer on the other hand will not see any of those characters.

WEKA allows use to adjust all this information via the StringToWordVector class.

Figure 3 shows an example of how the dataset looks like after data pre-processing.

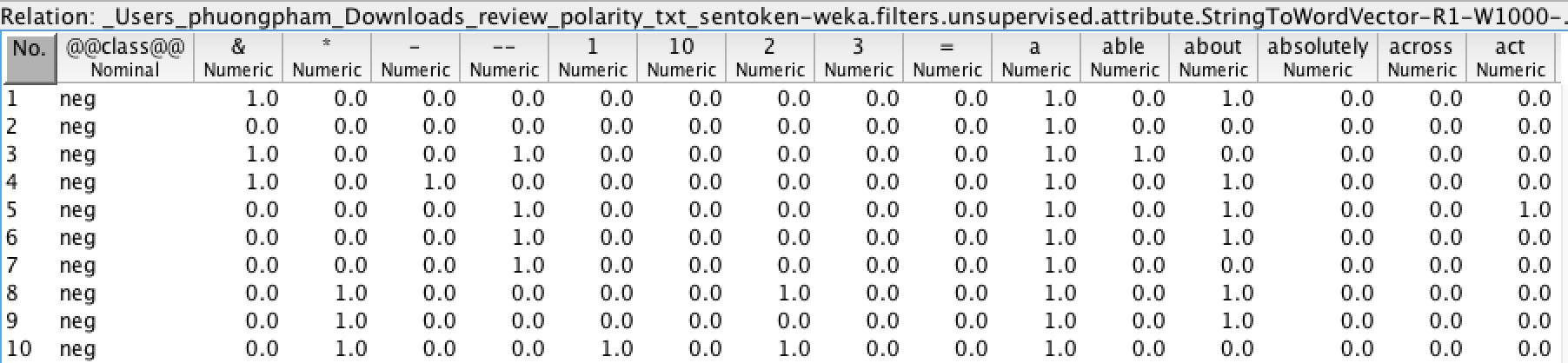


Figure 3 An example of how the dataset looks like after data pre-processing

**Model construction**

We will experiment the data set using cross-validation with k folds. The data set is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided.

We will use the pre-processed training set to build several models with the three well-known machine learning algorithm, including *MFNN*, *decision tree* and *Naivebayes* using Weka. Then we will compare the results generated by different models and conclude which machine learning algorithm works best for movie reviews sentiment analysis.

MFNN consists of an input layer, one more hidden layers and an output layer. Layers are connected in acyclic graph. Each layer is made up of computational elements called **neurons**. Neurons between two adjacent layers are pairwise connected, but neurons within one layer share no connection. No direct connection exists between input and output layer. Inputs are fed into the neurons making up the input layer. The outputs produced by this layer are weighted and passed simultaneously to the first hidden layer. This hidden layer outputs are again weighted and sent to an another hidden layer and so on. The weighted outputs of the last hidden layer are sent to the output layer, where the prediction for the given tuples will be produced.

* A decision tree is built as follows. Information gain is computed for each attribute. Attribute with the highest information gain will be selected as **root**. Examples are partitioned recursively based on selected attributes. The partitioning process stops when one of the following conditions is met:
* All samples for a given node belong to the same class
* There are no remaining attributes for further partitioning
* There are no samples left

Naivebayes assumes that attributes are conditionally independent. Which means the existence of one word does not affect that of an another. It will first calculate the probability that a document is positive and negative when it contained each particular attribute. It will then calculate the probability that document is positive and negative when it contained all attributes discovered by the data pre-processing step. The document belongs to the class where Naivebayes yields higher probability.

**Model evaluation**

We will use the testing set to evaluate the accuracy of each trained model. The statistical value of the accuracy is visualized by a *confusion matrix*, as shown in the table bellows.

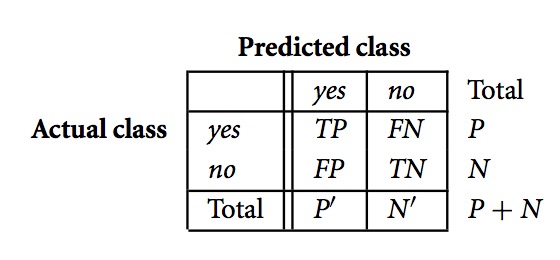


Table 1 Confusion matrix

Where TP is the number of true positive tuples, TN is the number of true negative tuples, FP is the number of false positive tuples and FN is the number of false negative tuples.

The accuracy or *correctly classify instances* of each model is calculated as follow.

The error rate or *incorrectly classify instances* is the negation of the accuracy.

**Experiment results**

Table 2 shows the comparison the accuracy of generated by different models.

|  |  |  |  |
| --- | --- | --- | --- |
| Method for choosing attributes | MFNN | Decision tree | NaiveBayes |
| wordsToKeep = 700; useStopList = False; normalizeDocLength = Normalize all data; minTermFreq = 1; tokenizer = AlphabeticTokenizer | 50.4438 % | 63.9015 % | **81.5485 %** |

Table 2 Comparison the accuracy of generated by different models

Our experiment shows that Naivebayes has the best accuracy which is **81.5485** %. It occurs when the number of kept words is 700. The use of stop list is discarded. The length of all documents are normalized so that the effect from number of times a word appear in a document is not affected by the length of a document. A word has to appear in a document more than 4 times in order to be considered as an attribute. Naivebayes algorithm also took the least time to train a model. Decision tree took a bearable time, whereas MFNN took a significantly long time to train a model.

**Discussion & conclusion**

Table 3 shows the accuracy of the experiments on NaiveBayes algorithm using different methods for choosing attributes. The experiment shows that the use of stop list actually lowers the accuracy of a model. The accuracy increases as the number of kept words increase, but only up to 700, after that, the increase of the number of kept words decreases a model’s accuracy. The use of Alphabetic tokenizer increases the accuracy. The accuracy also increases when the length of all documents are normalized. Finally, the minimum amount of time a word occurs in a document modestly affects the accuracy.

|  |  |
| --- | --- |
| Method for choosing attributes | Accuracy |
| wordsToKeep = 100; useStopList = False; normalizeDocLength = Normalize all data; minTermFreq = 5; tokenizer = AlphabeticTokenizer | 63.7506 % |
| wordsToKeep = 700; useStopList = False; normalizeDocLength = Normalize all data; minTermFreq = 5; tokenizer = AlphabeticTokenizer | 81.3977 % |
| wordsToKeep = 800; useStopList = False; normalizeDocLength = Normalize all data; minTermFreq = 5; tokenizer = AlphabeticTokenizer | 80.9452 % |
| wordsToKeep = 700; useStopList = True; normalizeDocLength = Normalize all data; minTermFreq = 5; tokenizer = AlphabeticTokenizer | 79.0347 % |
| wordsToKeep = 700; useStopList = False; normalizeDocLength = No Normalization; minTermFreq = 5; tokenizer = AlphabeticTokenizer | 80.6938 % |
| wordsToKeep = 700; useStopList = False; normalizeDocLength = Normalize all data; minTermFreq = 1; tokenizer = AlphabeticTokenizer | **81.5485 %** |
| wordsToKeep = 700; useStopList = False; normalizeDocLength = Normalize all data; minTermFreq = 100; tokenizer = AlphabeticTokenizer | 81.4982 % |
| wordsToKeep = 700; useStopList = False; normalizeDocLength = Normalize all data; minTermFreq = 5; tokenizer = WordTokenizer | 80.9452 % |
| wordsToKeep = 700; useStopList = False; normalizeDocLength = Normalize all data; minTermFreq = 5; tokenizer = NGramTokenizer | 77.5264 % |

Table 3 Accuracy of the experiments on NaiveBayes algorithm using different methods for choosing attributes

In conclusion, this project experimented the movie reviews from iMDb archive by building different classifying models using different algorithm and different ways of representing the experimental dataset. This project found that Naivebayes was the most appropriate algorithm for movie reviews classification. NaiveBayes model can reach to the accuracy of 81.3977 % within a rather short amount of time compares to the other two algorithms that was experimented in this project. There might be other combination of algorithm and method of representing the experimental dataset that yield a better accuracy. However, within the given amount of time, it was the best we could obtain by far. The results from this project could be adopted to develop a system that recommends users on which movie to watch based on the existing reviews.

**Reference**

[1] Kang H., Seong Y. and Dongil H. Expert Systems with Applications 39 (2012) 6000–6010. Senti-lexicon and improved Naïve Bayes algorithms for sentiment analysis of restaurant reviews

[2] IMDb's archive for the [rec.arts.movies.reviews](news:rec.arts.movies.reviews) newsgroup. <http://www.imdb.com/reviews/index.html>. Accessed 14 Oct 2016.

[3] Movies review data. <http://www.cs.cornell.edu/People/pabo/movie-review-data/>. Accessed 14 Oct 2016.