

# **MACHINE LEARNING**

**CS 6375.002**

**FALL 2013**

## **PROJECT REPORT**

**12/13/2013**

---

### **SENTIMENT CLASSIFICATION OF REVIEWS IN IMDb DATA**

---

**- SUDARSHAN SUDARSHAN**

**- sxs121433**

## 1. ABSTRACT

Sentiment analysis (classification) is the set of techniques that allows detection of emotional content in text. In this project, I have implemented a series of classifiers, namely, Naïve Bayes, Logistic Regression and Support Vector Machines, to distinguish between positive and negative sentiments on the IMDb movie reviews dataset. I also compare the results obtained by these classifiers with that of traditional topic-based classifier.

## 2. INTRODUCTION

The amount of information being gathered is growing exponentially, as part of the effort to better organize this information for users; researchers have been actively investigating the problem of automatic text categorization.

Text categorization (a.k.a. text classification) is the task of assigning predefined categories to free-text documents. It can provide conceptual views of document collections and has important applications in the real world. For example, news stories are typically organized by subject categories (topics) or geographical codes; academic papers are often classified by technical domains and sub-domains; patient reports in health-care organizations are often indexed from multiple aspects, using taxonomies of disease categories, types of surgical procedures, insurance reimbursement codes and so on.

Text categorization will help in retrieving the data easily and also to perform various analytics on the data to get deep insights. Ex: An about to be launched product's negative and positive reviews. These insights will help a company to improve their profits by coming out with various improved versions of the product or with a new product as a whole.

Recent years have seen rapid growth in on-line discussion groups and review sites (e.g., the New York Times' Books web page) where a crucial characteristic of the posted articles is their sentiment or overall opinion towards the subject matter. Categorizing these reviews automatically will help the company to predict the sales, profit margins and also the launch the product at a larger scale.

## 3. EXPERIMENTAL SETUP

In this project, I have implemented the Sentiment Classification using various Machine Learning Methods to solve the problem of text categorization [1]. And, also analyze the effects of various methods and their effectiveness.

### 3.1 Dataset

My dataset consists of 1400 IMDb movie reviews (files) [2] which I obtained from the Cornell University's NLP website [3]. The dataset has been preprocessed and equally divided into positive reviews and negative reviews by the NLP Group.

For my Experimentations, I have divided the original dataset into 3 different datasets, which differ in the number of training and testing samples. I have done this to evaluate the results obtained on using more (high) training samples with that of less (low) training samples. Table 1, depicts the division. I have also considered Bag-of-Words approach for all calculations.

	# of Training Examples (split %)	# of Testing Examples (split %)
<b>Dataset 1</b>	490 positive & 490 negative (70%)	210 positive & 210 negative (30%)
<b>Dataset 2</b>	630 positive & 630 negative (90%)	70 positive & 70 negative (10%)
<b>Dataset 3</b>	280 positive & 280 negative (40%)	420 positive & 420 negative (60%)

Table 1: Dataset Distribution

### 3.2 Baseline System Results

Sentiments in reviews are usually expressed using certain words, for example a user review can be: Lord of the Rings' It was the **worst** movie I have ever seen till date. Here, "worst" says that the user review is negative. Therefore, we can build a baseline system, using certain words which will tell you whether the reviews are positive or negative. This baseline system will act as minimum threshold, which the classifiers have to cross to be an efficient and reliable model.

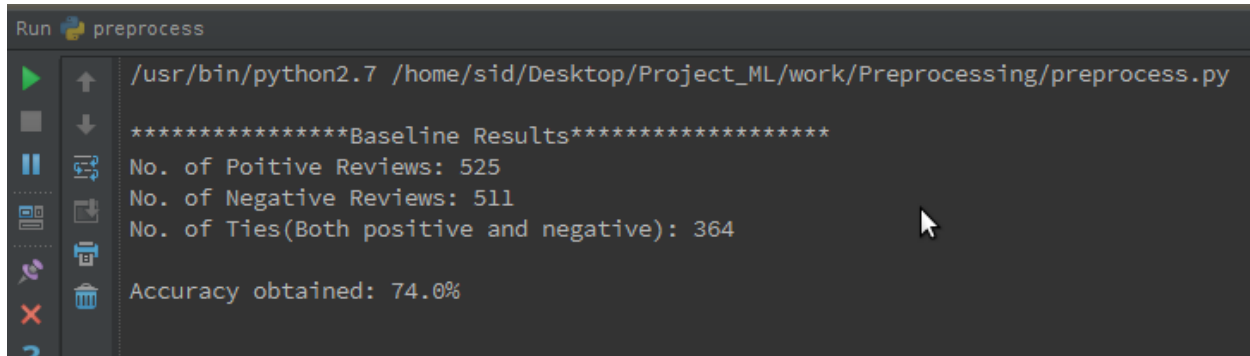
To obtain the baseline system, I took 14 words, that would be indicative of a positive movie review, and 14 words that would be indicative of a negative review. The words are shown in Table 2.

<b>Positive words:</b>	dazzling, brilliant, phenomenal, excellent, fantastic, mesmerizing, spectacular, awesome, moving, exciting, wonderful, superb, love, great
<b>Negative words:</b>	terrible, awful, unwatchable, cliched, sucks, boring, stupid, slow, bad, worst, stupid, waste, boring, pass

Table 2: Words used to obtain Baseline System Results

To classify the review as either positive or negative, I counted the number of times the positive word or negative word has occurred. If the number of positive words is more than the number of negative words, I have assigned the review as POSITIVE. I have assigned the review as NEGATIVE if it contains more negative words than positive words. If the number of positive words is same as that of negative words I have assigned it as TIES.

The results or accuracy obtained by using this system is shown in Figure 1. These results were on all the 1400 reviews, irrespective of positive or negative tag.



```
Run preprocess
/usr/bin/python2.7 /home/sid/Desktop/Project_ML/work/Preprocessing/preprocess.py
*****Baseline Results*****
No. of Poitive Reviews: 525
No. of Negative Reviews: 511
No. of Ties(Both positive and negative): 364
Accuracy obtained: 74.0%
```

Figure 1: Baseline System Accuracy

Therefore, our classifiers should cross this accuracy = 74% to be considered better.

### 3.3 Classifier Results

For all the classifiers I have taken the data as it is, no stop word removal has been done.

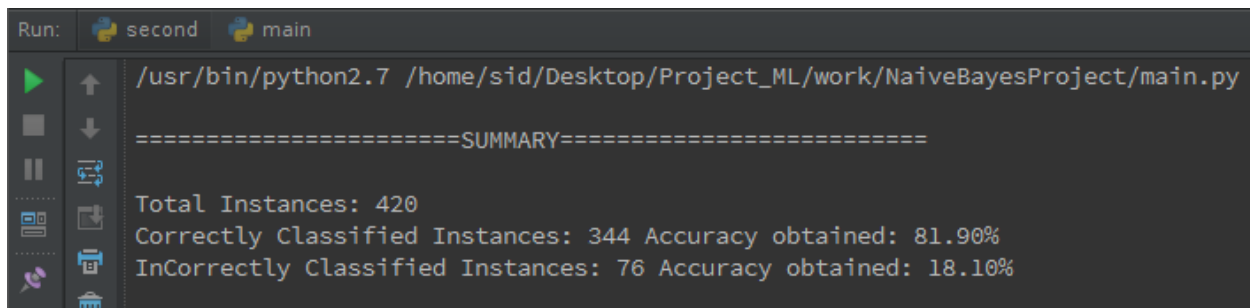
#### 3.3.1 Naïve Bayes Classifier (Multinomial Model)

I have implemented the Multinomial Naïve Bayes classifier as described in Information-Retrieval book [4] using Python programming/scripting language. I have taken into consideration all the tokens including special characters and made use of add-one Laplace smoothing.

The result obtained on different Datasets is shown in the Table 3, along with the Reference Figure label which points to the screen-shot of the results.

	Accuracy	Reference Figure
<b>Dataset 1</b>	81.90%	Figure 2(a)
<b>Dataset 2</b>	85.71%	Figure 2(b)
<b>Dataset 3</b>	76.31%	Figure 2(c)

Table 3: Accuracy obtained using Naïve Bayes



Run: second main

```
/usr/bin/python2.7 /home/sid/Desktop/Project_ML/work/NaiveBayesProject/main.py
```

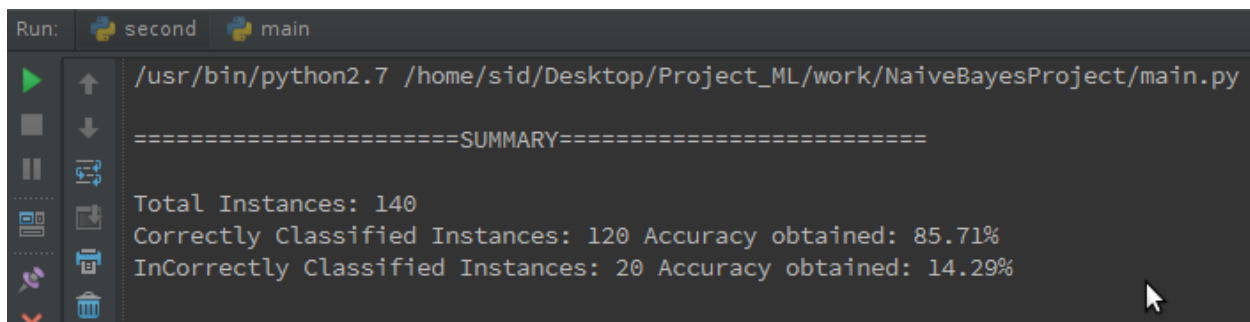
```
=====SUMMARY=====
```

```
Total Instances: 420
```

```
Correctly Classified Instances: 344 Accuracy obtained: 81.90%
```

```
InCorrectly Classified Instances: 76 Accuracy obtained: 18.10%
```

(a)



Run: second main

```
/usr/bin/python2.7 /home/sid/Desktop/Project_ML/work/NaiveBayesProject/main.py
```

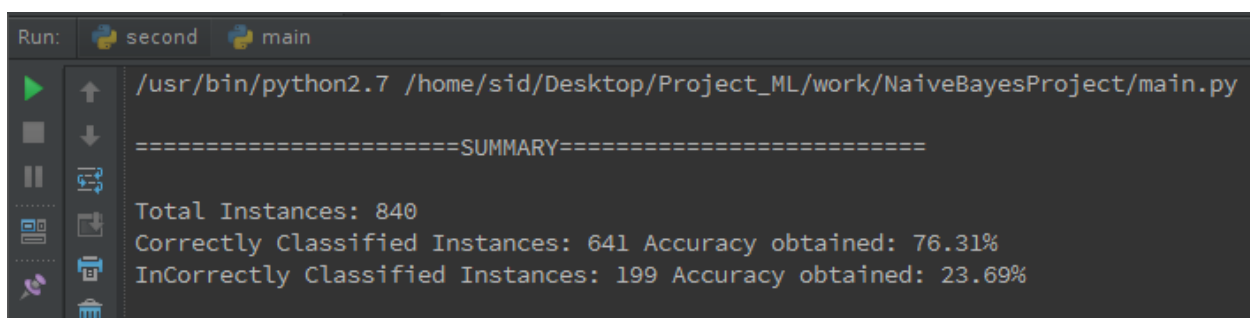
```
=====SUMMARY=====
```

```
Total Instances: 140
```

```
Correctly Classified Instances: 120 Accuracy obtained: 85.71%
```

```
InCorrectly Classified Instances: 20 Accuracy obtained: 14.29%
```

(b)



Run: second main

```
/usr/bin/python2.7 /home/sid/Desktop/Project_ML/work/NaiveBayesProject/main.py
```

```
=====SUMMARY=====
```

```
Total Instances: 840
```

```
Correctly Classified Instances: 641 Accuracy obtained: 76.31%
```

```
InCorrectly Classified Instances: 199 Accuracy obtained: 23.69%
```

(c)

Figure 2: Screen shot of results obtained by using Naïve Bayes Classifier

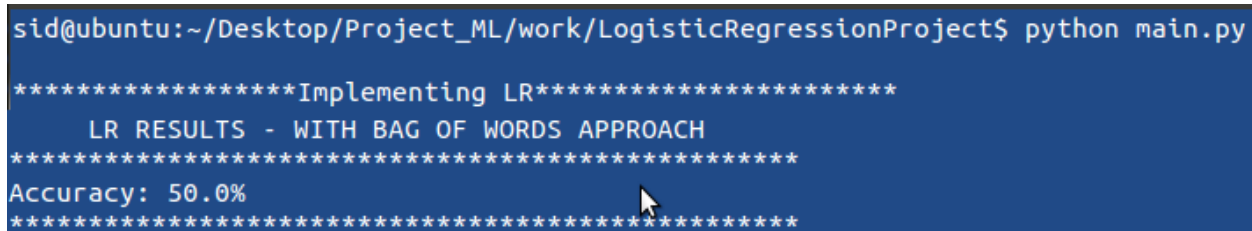
Dataset 1 and Dataset 2 were able to produce accuracies  $> 80\%$ . The Dataset 3, with less number of training samples (560) than testing samples (940), produced a result of 76.31%, which was also more than the Baseline System's accuracy of 74%.

### 3.3.2 Logistic Regression Classifier

I have implemented MCAP Logistic Regression algorithm [5] with L2 regularization using Python programming/scripting language. I have taken into consideration all the tokens including special characters.

The initial settings values are:  $\lambda = 0.001$ ,  $\eta = 0.0001$  and Iterations = 20 (Hard limit)

All the three datasets gave me an accuracy of 50%. As shown in Figure 3.



```

sid@ubuntu:~/Desktop/Project_ML/work/LogisticRegressionProject$ python main.py
*****Implementing LR*****
      LR RESULTS - WITH BAG OF WORDS APPROACH
*****
Accuracy: 50.0%
*****

```

Figure 3: Accuracy obtained by using Logistic Regression Classifier

Logistic Regression classifier accuracy obtained is less than that of the Baseline System accuracy. The reason for low accuracy might be due to insufficient data samples. The results might change with large datasets.

### 3.3.3 Support Vector Machines

Implementation of SVM was using Weka's LibSVM [6,7]. Weka takes data in .arff format. So, the text files needed to be converted to .arff format. I made use of Weka's TextDirectoryLoader API. The following command can be run in Weka's Simple CLI:

```
java weka.core.converters.TextDirectoryLoader -dir input_directory > output.arff
```

Once the text files are converted into .arff format the attribute text, which will have a String format, needs to be converted to the Vector format. We can do this using the Weka Explorer, once we have loaded the file, under *Preprocess* tab, we should choose *filter* -> *Unsupervised* -> *Attribute* -> *StringToWordVector* and click on Apply (with default values). Once the data has been converted into Vector format, go to *Classify* tab; choose the classifier *LibSVM* under *function*. You can also set the options of the LibSVM classifier by clicking on the tab next to *choose* under *classifier*. Make sure you have added the LibSVM classpath before starting the classifier process.

I have used C-SVC (Classification) SVMType for my experimentations. And, I have obtained the result using 4 different Kernel types, namely,

1. linear:  $u \cdot v$
2. polynomial:  $(\gamma u \cdot v + \text{coef0})^{\text{degree}}$
3. radial basis function(RBF):  $\exp(-\gamma |u-v|^2)$
4. sigmoid:  $\tanh(\gamma u \cdot v + \text{coef0})$

Kernel Type	Linear	Polynomial	RBF	sigmoid
<b>Dataset 1</b>	75.24%	71.43%	71.19%	47.38%
<b>Dataset 2</b>	76.43%	70.71%	70.71%	52.86%
<b>Dataset 3</b>	72.74%	69.64%	66.43%	53.45%

Table 4: Accuracy obtained by using SVM classifier

The results obtained by using SVM on the three datasets are shown in Table4. The linear SVM classifier results were more of an interest for my experimentation. The results obtained by Dataset 1 and Dataset 2 crossed the Baseline System Results. But, the results obtained for the Dataset 3, were disappointing for the fact that it didn't cross the baseline system result of 74%. The reason for this might be the No. of examples used for training (560) and the testing was done on a much larger dataset (940).

## 4. EVALUATION

The main aim of dividing the original dataset into three datasets was to analyze the impact of the number of examples used for training. By my experimentation results, we can clearly see that more the training examples have given better accuracy. But, how many examples might be sufficient? There is no answer for this question, at least as of now! – There is lot of learning going-on on Machine Learning.

Among the three classifiers, the Naïve Bayes classifier was better as it gave accuracies >80% when a reasonable amount of training data is provided. Although, SVM's are considered to give more accurate results than many classifiers, in my experimentation, the results were not so good as compared to Naïve Bayes. Logistic Regression, as always, was the worst performer among the 3 classifiers. Although my Machine Learning classifiers were able to produce better results than the Baseline System, when compared to the actual dataset classification (classified as positive and negative) they did not produce good results.

## 5. CONCLUSION

Sentiment Classification is one of the most important processes that many companies and researchers follow these days. In the era of Big Data, where exponential amounts of data are being gathered every day, the need for automatic text classification has increased and Machine Learning is a solution for this. In this project, I tried to implement various classifiers to classify the text data based on sentiment analysis. The results obtained were better than the Baseline System, where humans classified the reviews based on certain keywords. I have also analyzed the effects of choosing the right amount of training samples on the accuracy.

## REFERENCES:

- [1]. "Thumbs up? Sentiment Classification using Machine Learning Techniques", Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan, Proceedings of EMNLP, pp. 79--86, 2002
- [2]. <http://www.imdb.com/interfaces>
- [3]. <http://www.cs.cornell.edu/people/pabo/movie-review-data/> Sentiment polarity datasets - polarity dataset v0.9
- [4]. <http://nlp.stanford.edu/IR-book/pdf/13bayes.pdf> (see Figure 13.2)
- [5]. Logistic Regression Notes, Dr. Vibhav Gogate, The University of Texas at Dallas
- [6]. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.
- [7]. <http://www.cs.waikato.ac.nz/ml/weka/>