**Sentiment Analysis of Yelp Reviews – Naïve Bayes Classifier**

Nephi Calvin Bandela

**Introduction:**

Sentiment Analysis is a process of analyzing a piece of text and determining the opinion of the author. This opinion can be classified into different types. This project evaluates 2 types of classification of the text:

1. Classifying the text as positive or negative
2. Classifying the text into one of the ratings in the range [1,5]

The advantages of Sentiment Analysis are immense with the rise of internet and the people posting their feedback on products, places etc., Businesses look forward to understand this feedback to act upon their backlogs with ease and hence Sentiment Analysis comes to the rescue.

As humans we can easily predict to an acceptable amount of accuracy by reading a text content if it is positive or negative. The emphasis of this project is to emulate this behavior in machines and hence Artificial Intelligence comes into play. The project involves sufficient amount of learning (using the training data set) which is a major task of Machine Learning. Also, we are dealing with the interactions between a machine and human language which is the major area of Natural Language Processing. Hence Sentiment Analysis is an interesting area of Artificial Intelligence encompassing the branches of Machine Learning and Natural Language Processing.

**Background:**

Sentiment Analysis and Opinion Mining has recently gained a huge interest in research due to the rise of Machine Learning methods in NLP and Information Retrieval and also the availability of wide range of datasets with increase of review aggregation websites [1]. Most of the research on sentiment analysis tend to capture the effectiveness when classifying the text into either of [positive, negative] with use of algorithms like Naïve Bayes or Support Vector Machines [2]. The goal of this project is to find if it is possible to achieve a similar amount of accuracy when predicting the rating within a range [1,5] of a review. The problem here lies in the fact that prediction based on binary classification is easy whereas with multiclass[5-class] classification it tends to be hard because of the fact that the rating is subjective to the individual.

**Motivation:**

The Yelp Dataset Challenge [3] has a Natural Language Processing category which involves guessing the review’s rating from the text alone. This is bit of a challenge even for a human because review rating is subjective to the individual and differ from person to person but categorizing a review as either positive or negative on the other hand is an easy task. Even machines perform well in predicting this polarity(positive/negative) compared to predicting a rating within a range of [1,5]. A rating of 4 to a particular review by an individual might be a rating of 5 to a different individual. Correctly predicting the rating is challenging to humans itself and the goal of this project is to understand how accurate can a machine predict the rating of a review and what are the factors that improve the accuracy.

To better understand this process the Naïve Bayes Text Classification technique, a supervised learning algorithm is implemented and is evaluated by benchmarking with other algorithms.

**Problem Statement:**

Any text classification problem requires proper training of the classifier and require feature extraction as the main step [4]. Furthermore, the goal of this project is to compare the accuracy of prediction between 2-class [positive, negative] and 5-class [1,2,3,4,5] classification. Binary classification on one hand tend to be easy because containing a single positive word might lead to the review being classified as positive while on the other hand Multiclass classification tends to be harder as it cannot correctly capture the difference between a 5-star rating and a 4-star rating just by examining the amount of positive words and negative words [5].

**Methodology:**

The AI Machine Learning algorithm implemented in this project is:

* Naïve Bayes Classification – Supervised Learning

Naïve Bayes Classifier:

The Naïve Bayes Classifier is based on the following formula:

Posterior Probability = (Prior \* Likelihood) /Evidence ~ Prior \* Likelihood

In the current context the above formula can be described as:

The probability of class given a review is given by product of prior of the class and likelihood of the review belonging to that class which can be represented as the following:

p (class | review) = p (class)\* p (review | class)

where class is one of [positive, negative] or [1,2,3,4,5]

We calculate the posterior probability for each of the classes and assign the review to a class of highest posterior probability

Progress so far:

1. The implementation of the Supervised Learning Algorithm – Naïve Bayes which can classify a review to one of the classes.
2. The same implementation as above but using the algorithms (SVM, Naïve Bayes) from the Scikit Library.
3. Benchmarking the above implementations

Problems encountered – Alternate Solution:

1. Representation of data from the training set that needs to be fed to the classification algorithm – used a hash map containing the feature set (bag of words).
2. Low accuracy in the initial implementation - improved the accuracy by removing the stop words.

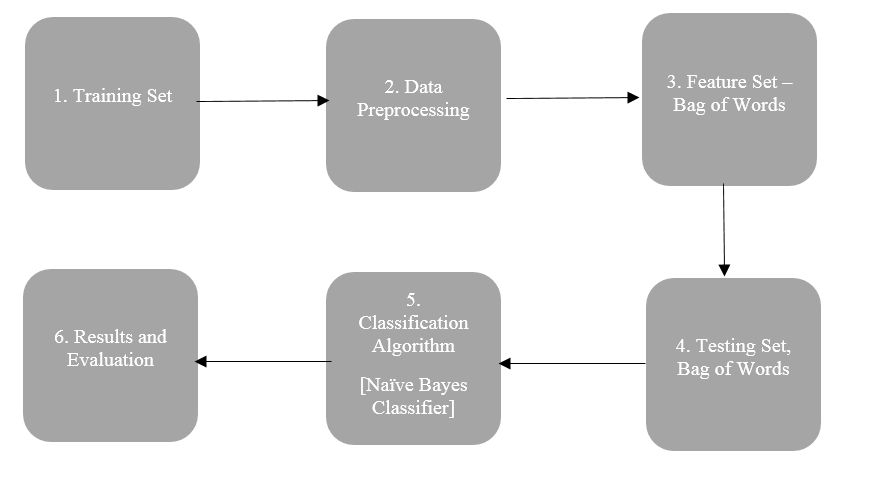
**Novelty:**

1. The preprocessing step handles tokenization, punctuation, stop words. Before passing the data to the Naïve Bayes Classifier the data is represented as “Bag of words”. Several preprocessing techniques perform differently and the current implementation works well for the ‘Bag of words’ feature set.
2. Flexibility to choose between the classifications i.e. User can choose to classify based on polarity [positive, negative] or within a rating of [1,2,3,4,5].
3. Naïve Bayes implemented from scratch and also compared with Naïve Bayes, SVM from Scikit Learn library.

**Analysis and Design of Project:**

Functional Requirements:

1. Dataset Selection: After researching through several datasets the Yelp dataset was chosen based on the fact that it provides diverse data and is a good choice for text classification.
2. Machine Learning Algorithm Selection: Given the deadline, after researching some of the supervised algorithms namely Naïve Bayes, SVM and Random Forests I chose to implement the Naïve Bayes algorithm since it guarantees good results.
3. Choice of programming language: Having worked with Java’s Weka library which includes the Machine Learning libraries the programming language Python is used to learn the language and also benchmark the algorithms using its popular Machine Learning library - Scikit learn.
4. Classification types: The dataset only provides data in the format of (review, stars) which is used to predict the rating within the range of [1,5] but in order to classify based upon the polarity(positive/negative) the dataset was reformatted to (review, polarity).
5. Feature Selection: Feature Selection is the most important task in text classification. A thorough research on how the sentiment analysis feature sets are selected is done and some of the popular methods are implemented in this project.
6. Training Set and Testing Set size: Any typical Machine Learning task includes proper division of training and test set. The typical size usually used is 85% training set and 15 % test set. This project uses 20,000 reviews to train and 2000 reviews to test.
7. **Design Architecture:**

****

1. **Results and Evaluation:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Classification  algorithm | Training Size, Testing Size  (reviews) | Classification  Type | Accuracy | Time Taken  (Seconds) | Precision | Recall |
| 1 | Naïve Bayes User Defined | 20000,  2000 | Positive or  Negative | 0.74 | 450 | 0.81 | 0.75 |
| 2 | Naïve Bayes User Defined | 20000,  2000 | [1,5] Stars | 0.36 | 938 | 0.38 | 0.36 |
| 3 | Naïve Bayes Predefined | 20000,  2000 | Positive or  Negative | 0.80 | 2.58 | 0.80 | 0.80 |
| 4 | Naïve Bayes Predefined | 20000,  2000 | [1,5] Stars | 0.42 | 2.34 | 0.35 | 0.43 |
| 5 | SVM Predefined | 20000,  2000 | Positive or  Negative | 0.89 | 2.67 | 0.90 | 0.90 |
| 6 | SVM Predefined | 20000,  2000 | [1,5] Stars | 0.52 | 3.60 | 0.51 | 0.53 |

**Note:** The user defined classification algorithm in the above results is the one where Naïve Bayes Classification Algorithm is implemented. The predefined algorithms are used from the Scikit Learn library.

1. **General AI evaluation:**

The methods used for the evaluation are Accuracy, Precision and Recall. Scikit learn has these methods built in its ‘metrics’ package.

The following values are computed by comparing the actual result with the predicted result.

Accuracy=(tp+tn)/(tp+tn+fp+fn)

Precision=tp/(tp+fp) = denotes the usefulness of the results

Recall=tp/(tp+fn) = denotes the completeness of the results

Where tp=true positive; tn=true negative; fp=false positive; fn=false negative

1. Positive Negative Classification:

The Naïve Bayes implemented has an accuracy of 74% when predicting the review as positive or negative. The precision value is 81% which tells us about “how useful the classified result is” and the recall value is 75% which tells us about “how complete the classified result is”.

When Naïve Bayes predefined algorithm from Scikit Learn Library is used we get an accuracy of 80%. The precision value is 80% and the recall is 75%

1. 5-Star Classification:

In this classification the user defined version gets an accuracy of 36% with 38% precision and 36% recall.

The predefined version has an accuracy of 42% with 35% precision and 43% recall.

Although the predefined version overtakes implemented version the implemented version still is helpful because the difference is just 6% and hence it is a successful classifier.

1. **Experiments and Results:**

The following are the results of Manual testing:

1. Positive Negative Classification: Correctly classified the review as “positive”
   1. Train Set: Labels: {'positive': 15787, 'negative': 4213}, Bag of Words Sample Representation: 'good': {'positive':9680, 'negative': 1911}
   2. Test Review: “excellent food great ambience loved it”
   3. Tokenization: ['excellent', 'food', 'great', 'ambience', 'loved', 'it']
   4. Remove Stop Words: ['excellent', 'food', 'great', 'ambience', 'loved']
   5. Prior Probability: positive prior = 0.78935, negative prior = 0.21065
   6. Likelihood: positive:6.30210411341e-06, negative:1.11004871498e-07
   7. Posterior Probability: positive:4.97e-06, Negative:2.33e-08

Note: 4.97e-06=4.97\*10^-6

* 1. Classification: “positive”

1. 5-Star Classification: Correctly classified the review with rating “4”
   1. Train Set: Labels: {'1': 2155, '3': 3097, '2': 2058, '5': 6692, '4': 5998}, Bag of Words Representation: 'good': {'1': 628, '3': 2539, '2': 1283, '5': 2676, '4': 4465}
   2. Test Review: “excellent food great ambience loved it”
   3. Tokenization: ['excellent', 'food', 'great', 'ambience', 'loved', 'it']
   4. Remove Stop Words: ['excellent', 'food', 'great', 'ambience', 'loved']
   5. Prior Probability: ‘1’=0.10775, ‘2’=0.1029, ‘3’=0.15485, ‘4’= 0.299, ‘5’=0.334
   6. Likelihood:1: 7.3e-09, 2: 5.4e-07, 3: 2.3e-06 ,4: 8.8e-06 ,5: 5.7e-06
   7. Posterior Probability: '1':7.8e-10,'3':3.6e-07,'2':5.6e-08,'5':1.9e-06,'4': 2.6e-06
   8. Classification: “4”

**The AI algorithm:**

Here is a detailed algorithm of the Naïve Bayes Classifier used in this project:

\*Test Set reviews: contains only the review with no class label, goal is to predict the class label

for each review in test Reviews:

\*Considering just two class labels

for each class Label in class Labels:

1.Calculate Prior: In the training set

Prior[class1] = class Label 1 Count/total count of reviews

Prior[class2] = class Label 2 Count/total count of reviews

2.Calculate Likelihood:

for word in review:

i.wordProbability[class1] = wordOccurrence/class Label1 count

\*Word occurrence in the training set; if word not in training return default

Probability which is very minute.

ii.wordProbability[class2] = wordOccurrence/class Label2 count

likelihood[class1]=word1Prob[class1]\*word2Prob[class1]\*…\*wordnProb[class1]

likelihood[class2]=word1Prob[class2]\*word2Prob[class2]\*…\*wordnProb[class2]

1. Calculate Posterior probability:

PosteriorProbability[class1] = prior[class1]\*likelihood[class1]

PosteriorProbability[class2] = prior[class2]\*likelihood[class2]

If(PosteriorProbability[class1]>PosteriorProbability[class2]): classify review as class1

If(PosteriorProbability[class1]<PosteriorProbability[class2]): classify review as class2

Thus we assign each review to a class based upon the posterior probability. If we are using the positive-negative classification, the classes are two namely positive and negative whereas using the [5-star] classification, the classes are 5 namely 1,2,3,4,5 respectively.

**Design and implementation tools (software and hardware):**

The project is tested successfully to run on a Windows 10, 64-bit i5 Processor, 8 GB RAM, 256GB SSD with python 2.7 and all the below mentioned libraries installed.

1. **Software:**

Programming Language: Python

Development Environment: PyCharm IDE

External Libraries: NLTK (Stop words), Scikit (Benchmarking using its SVM, Naïve Bayes)

Evaluation: Scikit Metrics (Classification Report, Confusion Matrix and Accuracy)

1. **Hardware:**

Processor: 32/64-bit

RAM: > 4GB

Hard disk: > 80 GB

**Discussion:**

**Observations and Interesting Features:**

The observations in preprocessing step:

1. Proper Tokenization: Tokenization is very important in text classification since words have diverse meanings and this meaning are very helpful in determining the polarity of the word
2. Removing Stop Words: Words like ‘this’, ‘then’, ‘he’, ‘were’, ‘to’ convey no meaning in Sentiment Analysis and has to be removed. Although ‘not’, ‘against’ are stop words they have not been considered as stop word since they are valuable words with negative connotation.
3. Removing Punctuation: Punctuations are of no value and hence the data should be preprocessed to remove every punctuation.

Performance in the user defined implementation is slow and can be improved by properly using the data structures.

Classification based on polarity [positive, negative] gives better results (74%) compared to the classification based on 5-classes [1,2,3,4,5] (36%). I observed the following to be the reasons:

1. From the training set in binary classification the data set is being split into only 2 classes and hence there are large set of examples to train on for each of these classes.
2. In multi class classification the data set is being split into 5 different classes and the training examples for each class are less and hence when we predict we get less accuracy

Another reason for binary classification better than 5-class classification:

Consider the review:

“So much fun Even for adults Especially for adults Interactive educational and tons of laughs The reptile show was great too”

The actual value in [positive, negative] classification is “positive”

The predicted value in [positive, negative] classification is “positive” which is a correct prediction

The actual value in 5-star classification is “5”.

The predicted value in 5-star classification is “4” which although might be wrong prediction but comes close to the actual prediction.

Even human beings cannot actually predict the above review as “5” or “4” because rating is very much subjective to each individual. Hence the algorithm works fine to simulate a human analysis of the text

Overall, Support Vector Machines- [89%,52%] beat Naïve Bayes Classifier- [74%,36%] by great accuracy with respect to classification on [positive/negative,5-star]. Some of the reasons for this are:

1. Naïve Bayes tends to work well with small data sets and Support Vector Machines work well with huge data sets.
2. Naïve Bayes assumes the conditional independence property which means in a review all the words are independent and does not influence the other whereas Support Vector Machines considers the review as whole.

**Conclusions:**

1. Binary Classification gives better results compared to 5-Class classification
2. Support Vector Machines perform better than Naïve Bayes
3. Naive Bayes user defined gives an accuracy of 74% while the predefined Naïve Bayes from Scikit Library gives an accuracy of 80% when performing binary classification.
4. Naive Bayes user defined gives an accuracy of 36% while the predefined Naïve Bayes from Scikit Library gives an accuracy of 42% when performing 5-class classification.

**Future Work:**

1. Currently developing a web based User Interface named ‘TeFy- Text Classify’ to predict classes based upon user entered text.
2. The implemented Naïve Bayes classifier can be improved by correcting the spelling mistakes in the words, removing emoticons.
3. This project can be extended to detect if there is any sarcasm in the review.
4. Rather than considering individual words for Naïve Bayes classifier the accuracy might be improved if two words are considered together.
5. The performance can be improved by testing on different data structures

**User's Manual:**

The user’s manual can be found at: [User Manual](https://drive.google.com/open?id=0BwFBoLIlHEtleUtWMXNYSFkxVm8).

**References:**

1. Bo Pang and Lillian Lee, *Opinion mining and sentiment analysis*, Foundations and Trends in Information Retrieval, 2008.
2. Bo Pang, Lillian Lee and Shivakumar Vaithyanathan*, Thumbs up? Sentiment Classification using Machine Learning Techniques,* 2002.
3. *Yelp*, https://www.yelp.com/dataset\_challenge.
4. Alexander Pak and Patrick Paroubek, *Twitter as a Corpus for Sentiment Analysis and Opinion Mining,* 2010.
5. Bo Pang and Lillian Lee, *Seeing stars: Exploiting class relationships for sentiment*

*categorization with respect to rating scales*, 2005.

**Project by:**

Nephi Calvin Bandela, Computer Science Graduate Student, Northeastern University

**Work Allocation:**

1. Sentiment Analysis using user defined Naive Bayes implementation

2. Sentiment Analysis using predefined Naive Bayes, SVM implementation (Scikit library)

3. Evaluation of the above 1 and 2

**Acknowledgement:**

1. Scikit Learn Library for the Naïve Bayes, SVM, Metrics; NLTK for Stop words
2. [Sebastian Raschka](http://sebastianraschka.com/Articles/2014_naive_bayes_1.html) for the detailed explanation on Naïve Bayes.
3. <https://en.wikipedia.org/wiki/Naive_Bayes_classifier> for the Naive Bayes overview.