ILS Z534 Project Report

YELP DATASET CHALLENGE

Group members

Giridhar Gomatom (ggomatom@indiana.edu)
Rohit Ingle (rpingle@indiana.edu)
Manasa Patibandla (mpatiban@indiana.edu)
Megha Redkar (msredkar@indiana.edu)

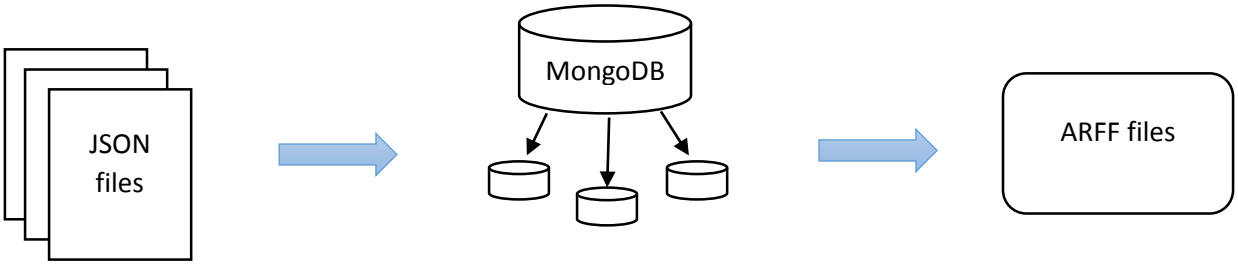
Under the guidance of
Prof. Xiaozhong Liu
at
Indiana University Bloomington

INTRODUCTION

The yelp dataset challenge involves data from Phoenix, Las Vegas, Madison, Waterloo and Edinburgh consisting of approximately 42153 businesses where users have provided their tips and reviews for these businesses. The challenge here is divided into 2 tasks. The first task expects these reviews to be classified appropriately by using the reviews that are given to them by multiple users. The second task involves rating a particular business using the tips and reviews available for each of the businesses.

PREPROCESSING

The original Yelp dataset is provided in JSON format. Here, we stored the information present in JSON files as Mongo DB collections and then use mongoDB information for these files which helps us to retrieve necessary information and ignore the ones that are not relevant to the tasks. These files are later converted into ARFF file format since the machine learning tools that are used for this challenge takes the input files in ARFF format.

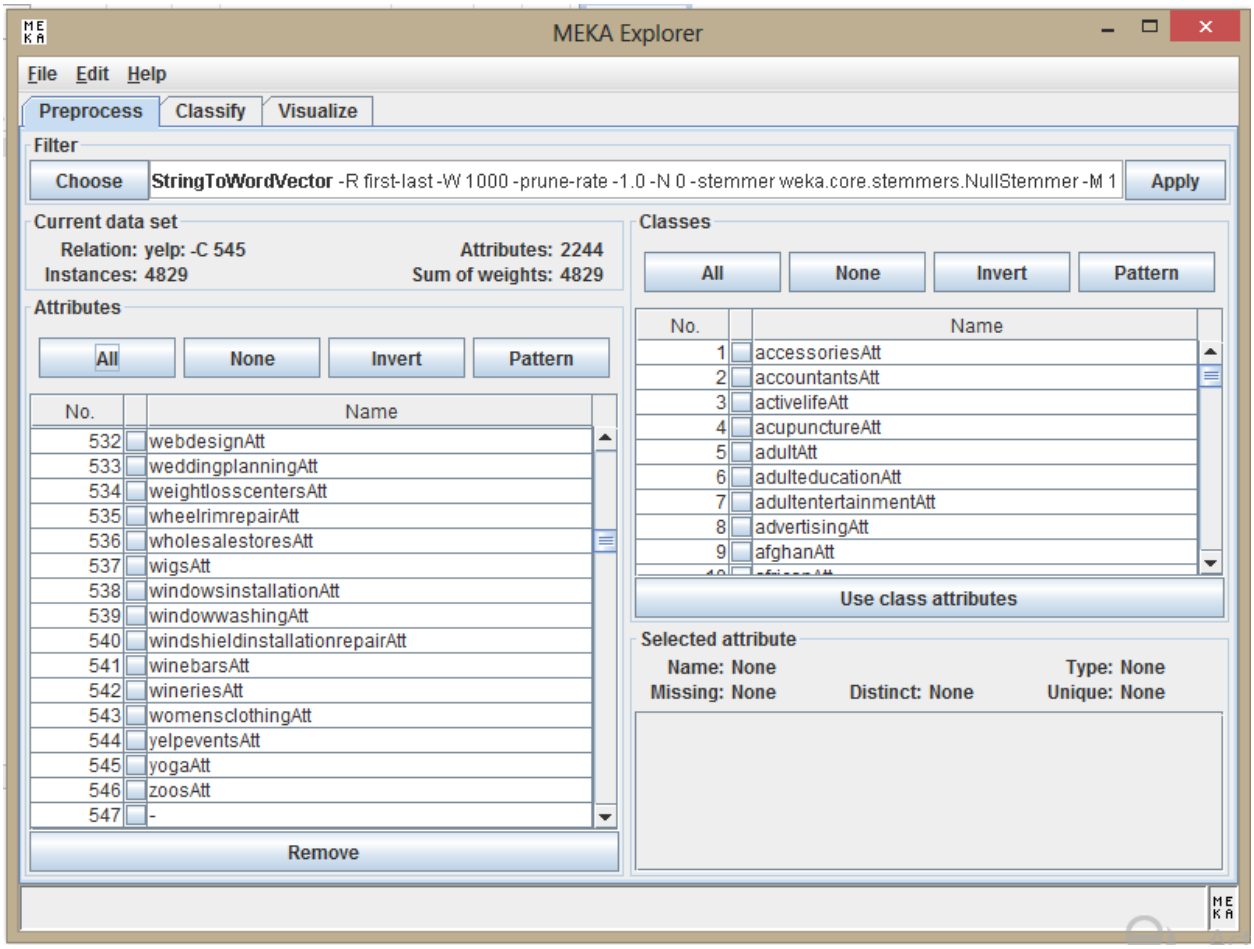


The above image explains the transition of the JSON files to ARFF files which are then further divided into Training set and Testing set by MEKA or WEKA.

TASK 1: Predicting categories from review text

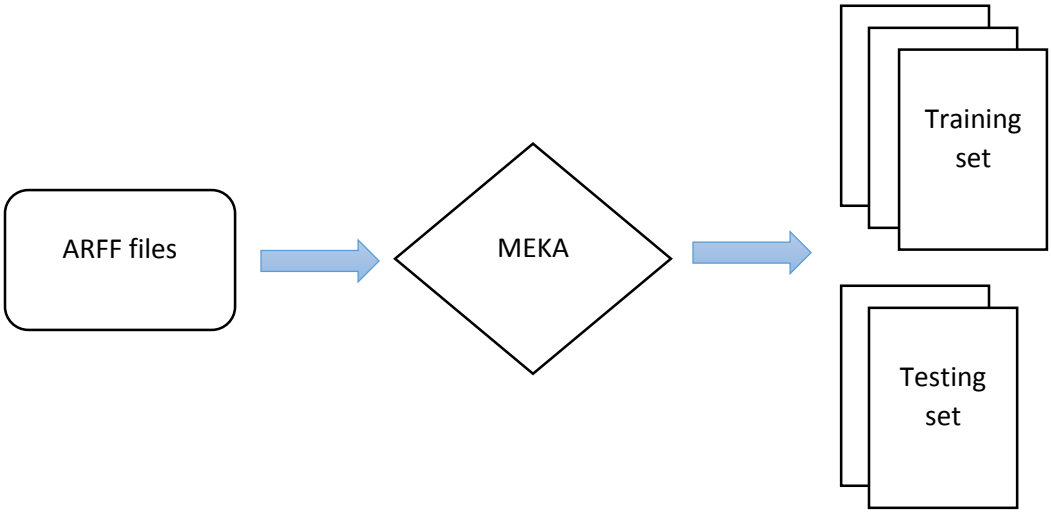
Algorithm:

The ARFF files obtained from the above process will be given as input to the MEKA tool. MEKA is a multi-label extension to WEKA which is a machine learning tool. For testing purposes, we perform feature selection of different types on the ARFF files like parts of speech tagging where the nouns will be extracted from the reviews and written into a new file, files where stemming is performed, files with stop word removal, etc. These files are then browsed through the WEKA interface and then the string to word function is selected to convert the string into words.



We use the Naïve Bayes classifier to categorize the reviews into classes. These files are classified using binary relevance technique where each review is a separate instance and the existence of that review in the class is denoted by binary digit 1 and absence is denoted by binary digit 0.

The training set and the testing set is split using MEKA itself and here, we define the split percentage as 80 percent.



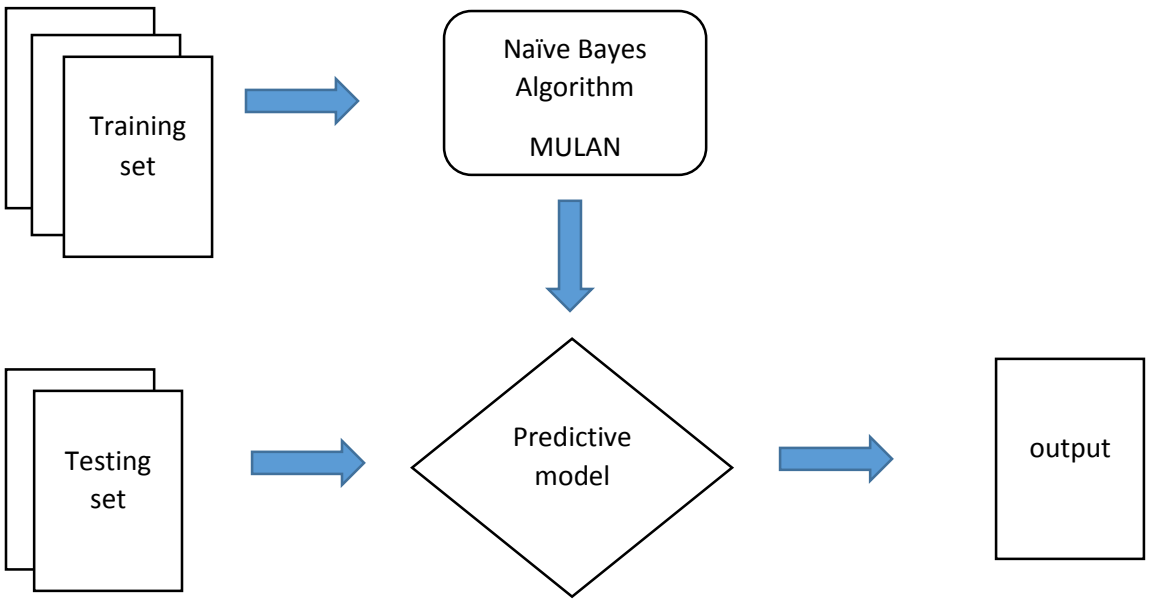
Naïve Bayes classifier is used here to predict the labels of the instances in the testing set, the input given to this algorithm is a training set that has instances with appropriate labels. The algorithm calculates the probability of each class occurring in the training set:

P (class) = No. of reviews that is labelled with the class / Total no. of reviews

The algorithm also calculates the probability of each word occurring in every class:

P (word|class) = [count (word, class) + 1] / count (class)

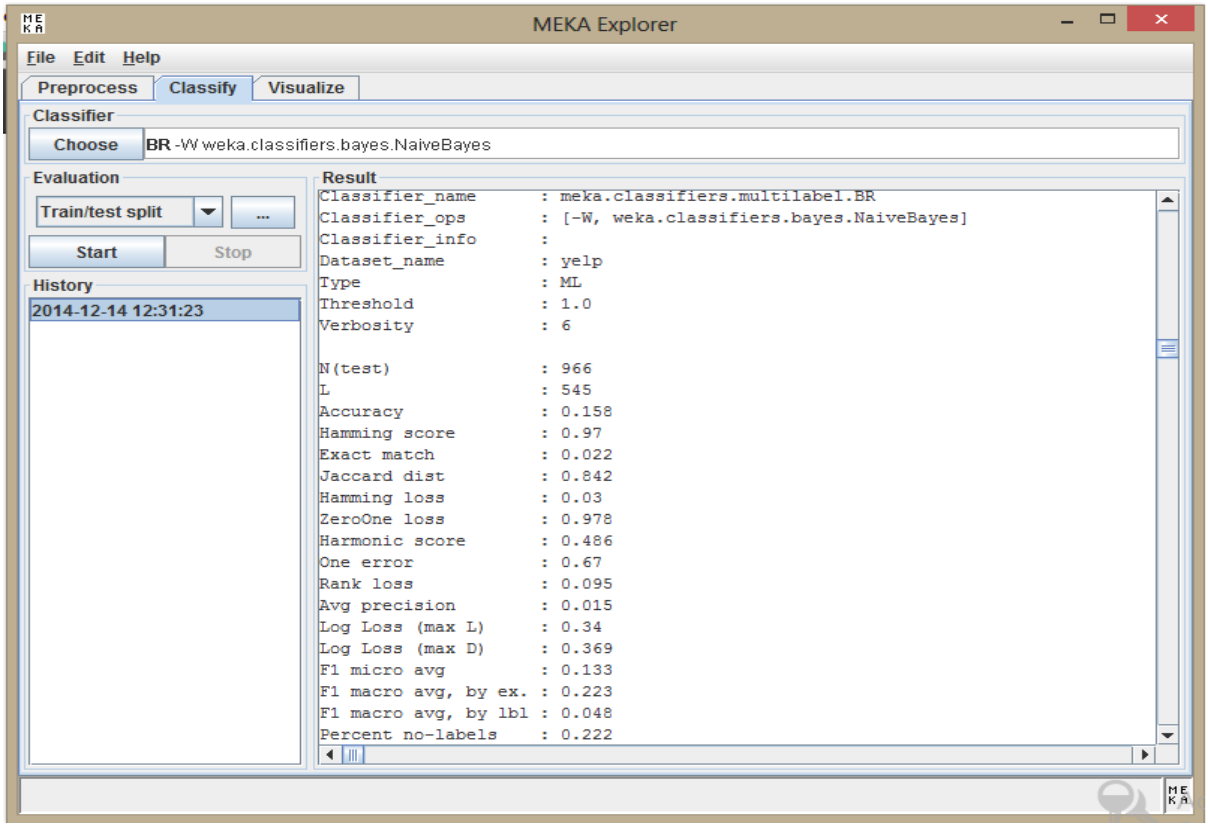
These probabilities are then used to classify the instances of reviews in the testing set.



Evaluation Metric:

1) Hamming loss: Hamming loss can be defined as the number of labels predicted incorrectly out of total number of labels. The hamming loss is expected to be as low as possible for the testing set.
2) Hamming score: and accuracy: Hamming score can be defined as accuracy which is the number of correct labels divided by the predicted labels and the true labels.

Results:



The values of the evaluation metric can be observed from the above results of MEKA and then a bar chart can be deduced by observing those values.

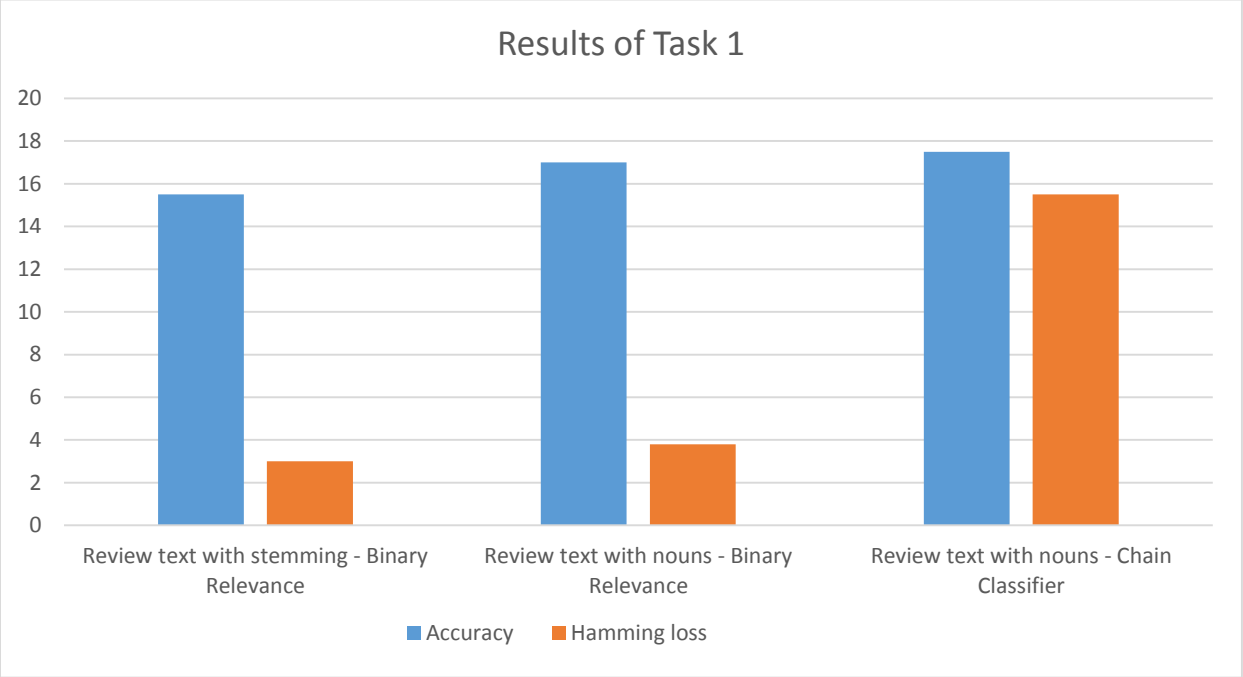
We have used 3 different datasets for testing:

1) Review text with stemming - Binary Relevance
   Here, the review text is filtered for stemming words and all occurrences of the same word in different forms in the review text are handled. Binary Relevance is used to represent

the review belonging to a particular class by 1 or 0. We observe that, the accuracy is relatively high and this approach gives us the smallest hamming loss.

2) Review text with nouns – Binary Relevance
   Here, the review text is filtered with parts of speech tagging and all word except the nouns are filtered out of the reviews from the dataset. The accuracy relatively increases than the previous approach but the hamming loss also shows a slight increase.

3) Review text with nouns – Chain Classifier
   The review text is the same as used in the 2nd approach but the chain classifier is where the output of the previous reviews are used to predict the output of the next set. The accuracy is the highest but the hamming loss is the highest as well.



TASK 2: Predicting User Ratings from Review text

Algorithm:

In task-2, we are performing the function of predicting the user ratings from Review Text.

We have adopted a supervised machine learning approach for this task. The algorithms used for classification are Multinomial Naïve Bayes and Linear Logistic Regression. Both these algorithms are known to perform well for text classification tasks and speed of execution is very fast.

For this task, we have worked only on the 'reviews.json' file which consists of business id, review and ratings. Since we are not concerned with the business id, we have used Mongo db to extract a file consisting of only reviews and their corresponding ratings. The ratings spilt is as follows:

- 5 Star ratings - 400k
- 4 Star ratings – 340k
- 3 Star ratings – 160k
- 2 Star ratings – 100k
- 1 Star ratings – 100k

Due to the vast size of the dataset and also memory constraints on our machine, we have performed random sampling on this large dataset and selected only a small subset from it.

Two datasets have been used for this task:

- Dataset-1 : 100000 instances (20000 each of 1,2,3,4 and 5 star ratings)
- Dataset-2 : 30000 instances (15000 each of 1 and 5 star ratings only)

Dataset-2 is used for Binary Classification. Using this dataset, our model classifies into either 1 or 5 star ratings. This is to test the performance of our model.

Feature Selection:

1. Stemming and Stopword Removal
2. NGrams – Unigrams, Bigrams and Trigrams
3. Part of Speech Tagging – Nouns, Adjectives, Adjectives + Verb

Tools Used:

- Environment – Eclipse
- Language – Java, Weka API
- Toolkits Used – Stanford NLP Toolkit, POS Tagger

Evaluation Metric:

We have used two metrics to evaluate our model:

1) Accuracy: It is a measure of how many instances from the testing set were predicted correctly.
2) RMSE: It is a measure of the root mean square error. The lower the RMSE, better is the classification. RMSE gives us a better estimate for our machine learning models. If the model classifies a review as 4-star instead of 5-star, it will be assigned a lower RMSE. If the model classifies a review as 1-star instead of 5-star, it will be assigned a higher RMSE.

We have adopted a 5 fold cross validation approach to obtain a more robust measure of the evaluation metrics.

Baseline:

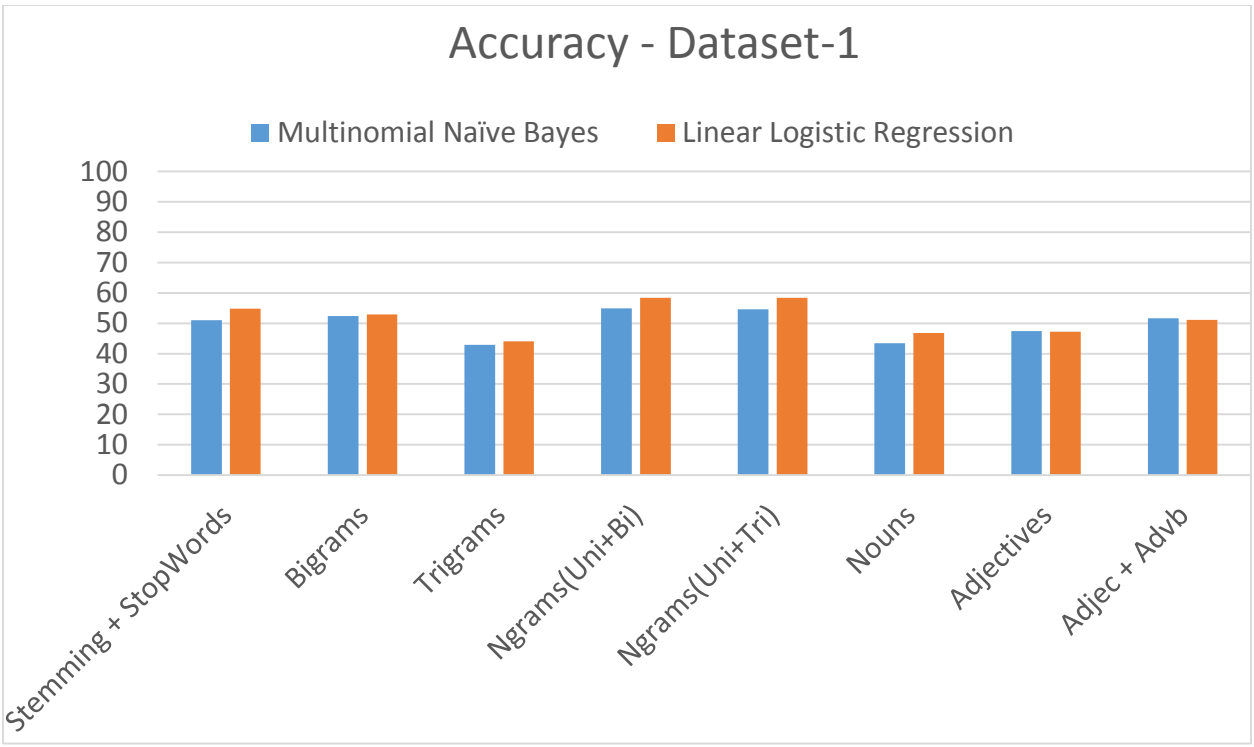
- Dataset -1: The baseline accuracy is set to 20% in this case since we can have classification into any one of the 5 classes.
- Dataset – 2: The baseline accuracy is set to 50% since this is a binary classification task. Hence the probability of a model of classifying it to either class is 0.5 (50%).

Results:

DATASET -1: Total Instances – 99999

Total Classes – 5 (One, Two, Three, Four, Five)

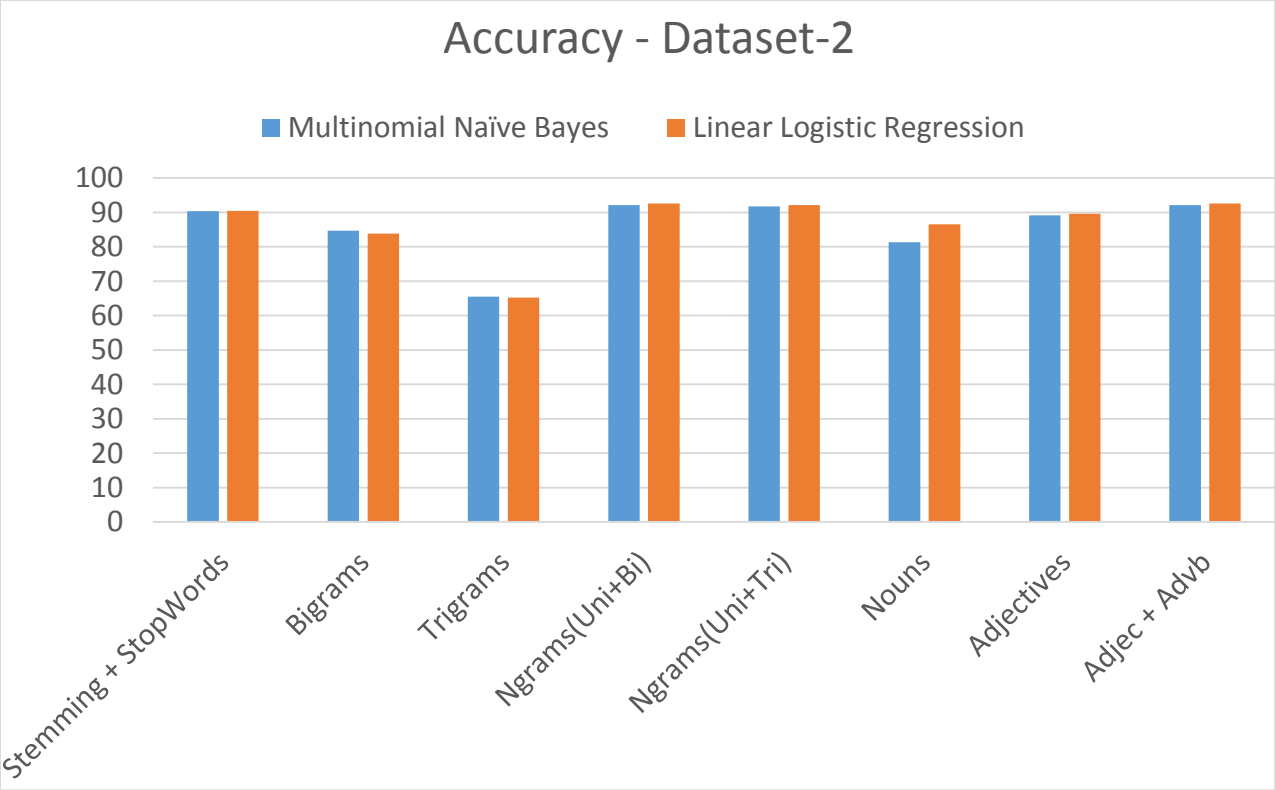
Type	Accuracy		Root Mean Square Error	
	Logistic Regression	Multinomial Naïve Bayes	Logistic Regression	Multinomial Naïve Bayes
Default	57.0202 %	52.4592 %	0.4146	0.3700
Stop Words + Stemming	54.8252 %	51.0219 %	0.4251	0.3710
Bigrams	52.3751 %	52.9416 %	0.4365	0.3722
Trigrams	42.9208 %	44.1869 %	0.4778	0.3744
Unigrams + Bigrams	58.3654 %	54.9183 %	0.4081	0.3745
Unigrams + Trigrams	58.3674 %	54.6861 %	0.4081	0.3762
POS Tag – Nouns only	46.8492 %	43.3922 %	0.4611	0.3871
POS Tag – Adjectives only	47.2716 %	47.4678 %	0.4593	0.3593
POS Tag – Adjective + Adverb	51.095 %	51.6935 %	0.4423	0.3512



DATASET -2: Total Instances – 30000

Total Classes – 2 (One, Five)

Type	Accuracy		Root Mean Square Error	
	Logistic Regression	Multinomial Naïve Bayes	Logistic Regression	Multinomial Naïve Bayes
Default	92.4672 %	92.1717 %	0.2745	0.2413
Stop Words + Stemming	90.4718 %	90.4137 %	0.3087	0.2631
Bigrams	83.8666 %	84.698 %	0.4017	0.3236
Trigrams	65.2329 %	65.5494 %	0.5896	0.4459
Unigrams + Bigrams	92.5834 %	92.1707 %	0.2723	0.2448
Unigrams + Trigrams	92.1046 %	91.7279 %	0.2810	0.2494
POS Tag – Nouns only	86.5752 %	81.3423 %	0.3664	0.3763
POS Tag – Adjectives only	89.6244 %	89.1916 %	0.3221	0.2730
POS Tag – Adjective + Adverb	92.6094 %	92.1727 %	0.2719	0.2413



Result Analysis:

1) We achieve accuracies of around 90% for the second dataset and around 55% for the first dataset.
2) Accuracy Gain:
    - Dataset 1 – 35% (55 – 20)
    - Dataset 2 – 40% (90 – 50)
3) For Dataset 1, the best performance in accuracy was achieved by the Unigrams + Trigrams model using Logistic Regression.
   The worst performance was achieved using the only Trigrams model. This is because splitting the text into trigrams and bigrams is a very costly operation and trigrams require a very large dimensional space. We had set the 'wordstokeep' in StringtoWordVector class as 2500 since we were running out of memory for more words. Increasing this parameter would help us achieve better accuracy with bigrams and trigrams.
4) In general, it is observed that Multinomial Naïve Bayes performs a better RMSE performance.
5) In general, it is observed that Linear Logistic Regression outperforms Multinomial Naïve Bayes in accuracy for this text classification task.
6) The performance decreased on using Stemming + Stopwords for both the classifier. This might be due to the fact that after stemming two distinct words get assigned the same stem. E.g. friendly and friendship – both stem down to friend. So if one classifier had

assigned friendly as one of the words for a specific rating and friendship to some other rating, after stemming the accuracy would decrease.

POS Analysis

- Amongst the POS tags used, the best performance was achieved using a combination of Adjective + Adverbs.
- This is because this helps to remove out all the unnecessary words and keeps only expressions and sentiment such as 'very tasty', 'indeed beautiful' etc.
- Using only Noun words is not that efficient for predicting user ratings since it will keep only object words in the review such as 'house' 'restaurant' 'Moby's' etc. (which is useful for predicting categories).

In the end, we conclude by saying that a decision between which classifier is better is not always a simple answer. There are a number of other factors that come into play such as type of dataset, size of dataset, feature selection etc. For our classification task, Logistic Regression outperformed Naïve Bayes.

It is usually observed that the Naïve Bayes performs better for small datasets. However, as the number of training examples grow, logistic regression will outperform Naïve Bayes. If the training data is scarce, one can expect Naïve Bayes to outperform logistic regression.

FUTURE SCOPE:

1) Try to run the same tests on a larger dataset and then check the performance of Naïve Bayes and Logistic Regression.
2) Increase the 'wordstokeep' parameter to a large value (say 10000) and then observe the improvement in performance of Ngrams (especially Bigrams and Trigrams). We need powerful hardware to carry this out.
3) Addition of weights to terms using TF-IDF.
4) Further enhance this model to build a fake review detection system based on a certain threshold.
5) Add Support Vector Machines and compare its performance as well.

CONCLUSION:

The Yelp dataset was used for 2 tasks where the ratings and the categories of the testing dataset were discovered using machine learning tools like MULAN and WEKA. Filtering techniques like stemming, stop words removal, parts of speech tagging, unigrams and bigrams were also used to filter the dataset. Stanford NLP was a toolkit that helped in making the filtering easier.