COMP 6721 Introduction to AI - Fall 2019

Project 2

40092843 Greeshma Sunil
greeshmasunil10@gmail.com

Appendix

1 Introduction

Machine learning has now become the answer to many practical real life problems. It has an extremely broad range of applications in such as spam/email filtering, decision making, categorizing data, sentiment analysis, medical diagnosis, image and pattern recognition and so on. Using this, we can create excellent algorithms by us of supervised/unsupervised learning techniques. This means that code need not be explicitly written. Text mining is an import application of this. There are a huge number of raw data stored in text format which can converted to useful information. It comes under Natural Language Processing (NLP). The challenge in this to create an optimal algorithm and to decide how to utilize this data to produce good results. In this era, Internet has been the major medium of communication and public content.

Because of this, it has become a major source for applying machine learning applications and therefore it enables us to have a large amount of data which can be used for training and testing the machine. Using machine learning and NLP, it is now feasible to predict the category of posts once training the model. This is very important in marketing strategies and other business applications.
Hacker News dataset which is taken form Kaggle. It is a technical website which features a huge amount of data in the form of stories posted by authors. These are comment and liked upon by thousands of visitors. In this project, we use this dataset to predict the category of post of each story. The Hacker news data set of 2018 is used for training our machine learning model and the 2019 dataset is used as the testing data. The purpose of this project is to understand practical application on machine learning and NLP by using classification algorithms and to experiment with the algorithms to obtain an optimal solution for this problem.

2 Technical Details

This project is coded in python 3. Python is one of the most popular languages used for Artificial Intelligence and Machine learning applications. This is due to its simplicity and vast libraries which are very helpful for the AI algorithm implementation and ultimately makes the tasks easier. The libraries required for this project are Numpy, Pandas, Nltk, Matplotlib and math.

3 Significance

The scope of this project is import for various real-world applications which are useful to evaluate the large amount of data stored in the internet. Classification of these unlabeled data would help easier evaluation and researches on the dataset.

4 Description of Architecture

4.1 Input Data Set

All the input files required for this project are stored in the Resources folder, The input data set is taken from Hacker News using Kaggle. The main data used for the project is 'hn2018_2019.csv'. The format of the model is as follows,

0	Object ID	Title	Post Type		Author	Created At	URL	Points	Number of Comments

Of this, we only consider the 'Title', 'Post Type' and 'Created At'. The project uses pandas to read from this csv file and separates the dataset based on 'Created At' into training and testing data. All the data in 2018 is used for training and 2019 data is used for testing. Since the training model is very large, it will take a good amount of time to complete the run. Due to this, processing this data requires code optimization techniques and memory and an organized plan of execution. The Post Type is of four types, 'story', 'ask_hn', 'show_hn' and 'poll'. It can be found from the dataset that most of the posts are of the type 'story', followed by 'ask-hn' and 'show-hn'. 'poll' category is the least observed Post Type. The project also requires a list of stop words which are used for eliminating certain words which are not useful for classification. This is store in stopwprds.txt

4.1.1 Training Dataset and Classification model

4.1.1.1 Pre-processing

The 2018 data in the file 'hn2018_2019.csv' is used as the training model for this project. This data is enormous and therefore requires lot of time for processing and creating the training model. For purposes of tests and demonstration, I have extracted chunks of data of different sizes and stored in various csv files. Before training the classification model, the data needs to be preprocessed. That is, we need to make changes to the 'title' values by removing unwanted symbols and words. Secondly, the title need to be lemmatized for better mapping of words of the same meaning not the words in the testing data. In this project, I have used WordNetLemmatiser form nltk.stem. the preprocessing of the titles are done in the function def **preprocess_model**(df): , where df is the data frame extracted from the input file for the year 2018. Once lemmatization is done, the titles are ready for tokenization.

4.1.1.2 Tokenization and Building model

For tokenizing the preprocessed titles, I have used nltk.word_tokenize from nltk. The counter() converts this into set of unique words and it's frequencies. The information required for each word is stored in 'word'class. This stores the frequency count of the word in each post type, total count and conditional probabilities of the word in each

class. The conditional probability is calculated using smoothed conditional probability and is given by p(wi/ci) = (frequency of wi in ci + smoothing value)/(total no.of words in ci + vocabulary size * smooth value). Since the amount of words are very large, this probability is very small. Therefore, we take log10 for the model. The output of the training model is stored in baseline-result.txt

4.1.2 Testing dataset and Niave Bayes Classifier

The testing model is extracted from the main dataset using the year as 2019. The testing model titles are also preprocessed in the same way as the training model before evaluation. The words in the testing model are matched with the training model to obtain the smoothed conditional values of the word for each class. A score value is calculated for each post type using the formula score of ci = prior probability of the class ci + sum smoothed conditional probabilities of all the words in ci for that title. The log of this value is taken to prevent numerical underflow and for better comparison. The predicted value generated is the value with the highest score value. This is called Niave bayes classifier.

4.2 Analysis of Results of Experiments on the Model

4.2.1 Base-line – Experiment 1

The baseline model uses 0.5 as the smoothing value as default. This is the model prior to any filtering of words. The results of training set are stored in model-2018.txt and the classification results of the test model are stored in baseline-result.txt in Output folder.

4.2.1.1 Analysis

The baseline model takes around 2.0 seconds to finish evaluating the model for ~titles. It can be seen that the model is quite accurate giving 70-99% accuracy rates. The true positive, true negative, false positive and false negative values can be analyzed as given in the console output. It can be noticed that is the words in the test set is not found in the training model, the classification is majorly based on the prior probabilities of the class. In our given dataset from Hacker News, it is noticed that most posts are from ‘Story’ post type. This means that, if the words are not found in the training data set, it is highly probable for this classifier to classify the post as ‘Story’ type. Since, most posts belong to ‘Story’ post type, this classification results as the right classification in ‘most’ cases.

4.2.1.2 Result

The result of the model is shown in the console in the form of various performance matrix related to the confusion matrix. We observe the accuracy, precision, Recall and F-value. This was calculated from the confusion matrix which was obtained from sklearn library functions. These measures give a good evaluation criterion for the model. Any value less than 0.5 can be bad classification results. From tests, it was observed that it Most of our results

showed good accuracy i.e. greater than 65%. It was noted that the classification showed better results for larger training set. But it also depended on the amount of words present in the test case that were found in the training set. For the input file sample.cv of ~100 samples, it showed 74.36% accuracy and some other files also showed accuracy rate >95%.

Sample	Accuracy	Precision	Recall	F1-Measure	Confusion Matrix
1.	0.794871	0.794871	0.7132616	0.794871	[[5 0 0] [2 1 0] [6 0 25]]
2	0.923076	0.92307	0.767025	0.923076	[[5 0 0] [2 1 0] [1 03 0]]
3	0.89743589	0.8974358	0.75627240	0.8974358	[[5 0 0] [2 1 0] [2 0 29]]
4	0.948717	0.94871	0.7777777	0.948717	[[5 0 0] [2 1 0] [0 0 31]]

```
--------------------------------------------------
Baseline model

Processing data..
Calculating probabilities...
Saving Model to file..
--------------------------------------------------
word count: 207 104 46 0
Vocabulary size: 317
--------------------------------------------------
Check file for output..

--------------------------------------------------
Testing Data...
--------------------------------------------------
Results
confusion matrix:-
 [[ 5  0  0]
 [ 2  1  0]
 [ 8  0 23]]
tn,fp,fn,tp : 0 2 1 1
Accuracy: 0.7435897435897436
Precision: 0.7435897435897436
Recall: 0.6917562724014337
F1 measure: 0.7435897435897437
This model is  74.36% accurate
--------------------------------------------------

Total elapsed time: 1.6 seconds
--------------------------------------------------
```

4.2.2 Stop word filtering – Experiment 2

Stop word filtering is performed over of the basemodel to remove certain words from the vocabulary which doesn't concern the model. The results of training set are stored in stopword-model.txt and the classification results of the test model are stored in stopword-result.txt in Output folder.

4.2.2.1 Analysis

It was observed that the removal of the stop words from didn't provide much difference in the classification results. The accuracy of this model was very good in most cases. In some case, the accuracy was low due to a possible factor that removal of words from the vocabulary might have reduced the number of matching words in the training set. Overall, it is a good choice to remove stop-words from the vocabulary.

4.2.2.2 Result

The result of the of the stopword experiment gave better accuracy, precision. F-measure and recall in majority of the cases. It was observed that evaluation results showed better results compared to the baseline model. For the input file sample.csv, this model showed 87.18% accuracy rate when compare to 74.36 accuracy rate. This shows that this experiment could increase the performance of the classification model.

```
Trying again with stop words...

-----------------------------------------------------
word count: 172 74 39 0
Vocabulary size: 268
-----------------------------------------------------
Testing Data...
-----------------------------------------------------
Results
confusion matrix:-
 [[ 5  0  0]
 [ 0  3  0]
 [ 4  1 26]]
tn,fp,fn,tp : 0 2 1 1
Accuracy: 0.8717948717948718
Precision: 0.8717948717948718
Recall: 0.946236559139785
F1 measure: 0.8717948717948718
This model is  87.18% accurate
-----------------------------------------------------
```

4.2.3 Word Length filtering - Experiment 3

Word length filtering removed the words that have length less than 2 and greater than 9. The results of training set are stored in wordlength-model.txt and the classification results of the test model are stored in wordlength-result.txt in Output folder.

4.2.3.1 Analysis

This experiment removes unwanted minute letters and Unicode symbols. If numbers and symbols were not removed in preprocessing, this step would help unwanted symbols and numbers. It was seen that this experiment showed an increase in accuracy rate compared to the previous models. The experiment was also repeated with other threshold values such as 2-12, 4-9, etc. It was seen that 2-9 was the best threshold for the experiment and hence the given 2-9 threshold was maintained in the project

4.2.3.2 Results

The results of this experiment showed better performance compared to the previous models. For the input taken as sample.csv, the accuracy rate for this was 92.31% which is much better than 87.18% in the previous experiment. Therefore, this experiment was taken to be a good choice for the model

```
Trying again with length filter...

-------------------------------------------------------
word count: 138 56 31 0
Vocabulary size: 210
-------------------------------------------------------
Testing Data...
-------------------------------------------------------
Results
confusion matrix:-
 [[ 5  0  0]
 [ 0  3  0]
 [ 2  1 28]]
tn,fp,fn,tp : 0 2 1 1
Accuracy: 0.9230769230769231
Precision: 0.9230769230769231
Recall: 0.967741935483871
F1 measure: 0.9230769230769231
This model is  92.31% accurate
-------------------------------------------
```

4.2.4 Infrequent word filtering - Experiment 4

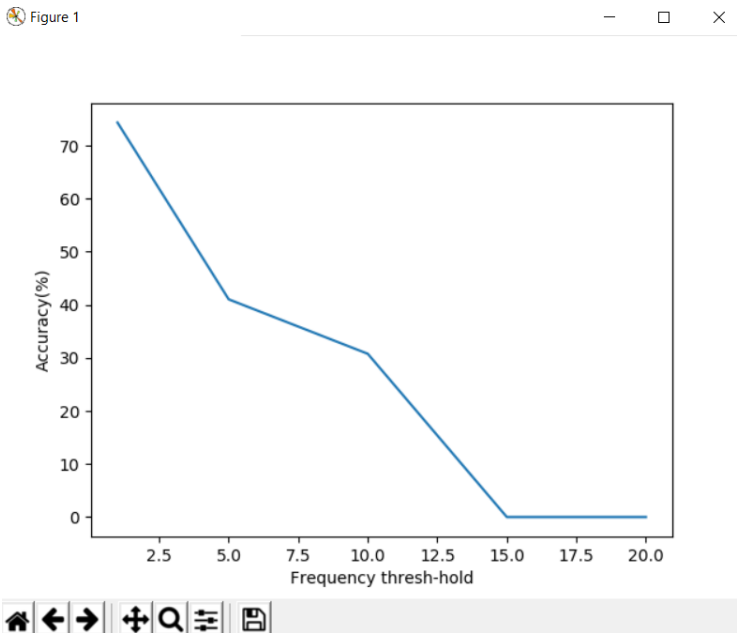
The words which has frequency <=1, frequency <=5, frequency <=10, frequency <=15, frequency <=20 was taken gradually in a for loop and the results were compared.

4.2.4.1 Analysis

It was found that, as the frequency threshold increases, the performance of the classifier was best for frequency<=1 and gradually decreases with higher threshold. In some cases, it increased to a peak and then decreased.

4.2.4.2 Results

In most cases, frequency <=15, frequency <=20 was a bad choice and decreased the performance up to 20% accuracy rate. This results helps us to decide on what kind of frequency threshold to choose when creatubg a classification model.



4.2.5 Smoothing – Experiment 5

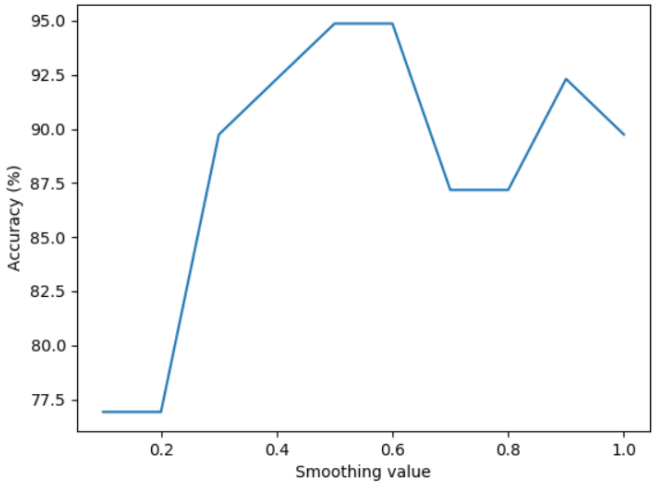
The smoothing value determines the value of the conditional probability of the words in the vocabulary. In this experiment, we test the performance for the testing model for smoothing value between 0 and 1 by gradually increasing it in a step size of 0.1.

4.2.5.1 Analysis

The variation of the performance according to the smoothing value was noted and recorded in the form a graph with y-axis-: accuracy percentage, x-axis- smoothing value. It was observed that the performance reaches its peak in the middle of the curve. The performance is lower for lower smoothing value and higher smoothing value. It was best for smoothing value = 0.4~0.9

4.2.5.2 Results

The smoothing value experiment was helpful for choosing a good smoothing value for the classification model.



4.3 Comparison on experiment Results

The result of the of the stopword experiment gave better accuracy, precision. F-measure and recall in majority of the cases. It was observed that evaluation results showed better results compared to the baseline model. For the input file sample.csv, this model showed 87.18% accuracy rate when compare to 74.36 accuracy rate. This shows that this experiment could increase the performance of the classification model when applied to the baseline model.

The results of this experiment showed better performance compared to the previous models. For the input taken as sample.csv, the accuracy rate for this was 92.31% which is much better than 87.18% in the previous experiment. Therefore, this experiment was taken to be a good choice for the model

In most cases, frequency <=15, frequency <=20 was a bad choice and decreased the performance up to 20% accuracy rate. Therefore, in order to get better performance than the previous experiments, we could make the frequency threshold as a small value such as 1~3.

It was observed that the performance reaches its peak in the middle of the smoothing value x accuracy curve. The performance is lower for lower smoothing value and higher smoothing value. It was best for smoothing value = 0.4~0.9. The baseline model and other models were performed using the default frequency 0.5, which was the best case in most of the cases. Therefore, this experiment didn't give much better results compared to the other values. But in some cases, when the smoothing value was higher than the baseline model, it provided very good performance.

5 Description of any difficulties

Since the complete 2018 training data set provided was very large in size, the classification takes a very huge amount of time to generate the training model. The training model for the project was performed by manual classification without any direct library functions such as MultinomialNB or GaussianNB which increased the size of the code. The project would've been simpler to evaluate and less complex if the model was binary i.e. has only two classes. There is more room in this project to improve the optimization and speed of the code