Text Classifier (Project 2)

COMP 6721

Nishant Saini[1] and Dhwani Sondhi[2]

[1] 40070734 nishantsainins@outlook.com

[2] 40083894 dhwanisondhi3@gmail.com

1 Introduction and Technical Details:

1.1 Description of the project:

The main motive of the project is to classify the user-submitted stories ("posts") with the help of the concept of natural language processing.

This project can be divided into several tasks:

- Reading data from a CSV file and dividing it into the training and testing data.
- Process and clean the data with the help of tokenization and lemmatization.
- Building the vocabulary and model based on it.
- Utilizing model to classify test data.
- Analyzing the effecting of stop-words, smoothing, word length filtering and infrequent word filtering.

1.2 Platform and Libraries:

- Jupyter Notebook[1]: It is an open-source web application that allows data cleaning, transformation, numerical simulation, data visualization, etc. This is used as a platform for our project.
- Pandas[2]: This is another open-source library easy-to-use data structures and data analysis tools. We used it to extract data from a CSV file and manipulate it.
- NLTK[3]: It is a library to work with human language data. We used this library to tokenize and lemmatize.
- Matplotlib[4]: It is a 2D plotting library. We used it for plotting plot after analyzing the data.
- Sklearn[5]: Simple and efficient tools for data mining and data analysis.
- NumPy[6]: It is used for scientific computation.

2 Data Analysis:

We are provided with a CSV file containing features like Object ID,Title,Post Type,Created At,URL, Points, and Number of Comments. For this project, we only focused on Title and Post Type because as we must implement Naïve Bayes these two are the most prominent ones.

3 Words in remove.txt:

In this section, we will discuss what are the words we removed in the baseline experiment during training.
First, we removed the punctuations by using regExpTokenizer, but those punctuations are not included in remove.txt. Then we started analyzing our model without removing any further words and by doing so we got an accuracy of 96.40%. Then we tried to remove words and analyzed what changes they produce on the accuracy. We found that by removing digits and words containing digits improved our performance to 98.70%.

4 Our Approach:

We implemented two models:

1. One from scratch without using the sklearn Naïve Bayes for all experiments
2. One with the help of sklearn Naïve Bayes for the baseline for comparison.

We used the second model as a reference for analysis in baseline cases.

5 EXPERIMENTS:

5.1 Baseline:

For this experiment we used the whole data set:
Without Sklearn Naïve Bayes(Our own implementation):

```
              precision    recall  f1-score   support

      ask_hn       0.81      1.00      0.89      5454
        poll       1.00      0.17      0.29         6
     show_hn       0.92      1.00      0.96      4903
       story       1.00      0.99      0.99    126852

    accuracy                           0.99    137215
   macro avg       0.93      0.79      0.78    137215
weighted avg       0.99      0.99      0.99    137215

Confusion Matrix
[[  5453      0      0      1]
 [     0      1      1      4]
 [     0      0   4903      0]
 [  1319      0    435 125098]]
Per class wise accuracy:
[0.99981665 0.16666667 1.         0.98617286]
Overall Accuracy: 98.71734139853514
```

Configuration:

- Delta = 0.5

Observations:

- As in the data, most of the examples are of “story”, so from the results, we can see that our model classifies the story testing data most accurately followed by “ask_hn”.
- As the “poll” type has very few training examples, so it is almost impossible for the model to classify it correctly. From the results, we can confirm it with the values of poll(precision: 0, recall: 0, f1-score: 0, accuracy: 0).
- One more thing we can see that mostly misclassification happened due to class story, as most of the words are present in class story hence it tends to bias the result.

With Naïve Bayes (by using inbuild Naïve Bayes):

```
                  precision    recall  f1-score   support

         ask_hn       0.91      0.99      0.95      5454
           poll       0.00      0.00      0.00         6
        show_hn       0.95      0.95      0.95      4903
          story       1.00      0.99      1.00    126852

       accuracy                           0.99    137215
      macro avg       0.71      0.74      0.72    137215
   weighted avg       0.99      0.99      0.99    137215

Confusion Matrix
[[  5418      0      0     36]
 [     2      0      0      4]
 [    42      0   4675    186]
 [   484      0    270 126098]]
Per class wise accuracy:
[0.99339934 0.         0.95349786 0.99405607]
Overall Accuracy: 99.25372590460226
```

Configuration:

- Delta=0.5

Observation:

- Here we can see the accuracy of sklearn Naïve Bayes. It managed to classify polls also.
- Also, misclassification reduced greatly.

5.2 Stop-word Filtering:

In this experiment, we used the whole dataset as in baseline, along with that, we introduced stop words.

```
              precision    recall  f1-score   support

      ask_hn       0.95      1.00      0.97      5454
        poll       0.00      0.00      0.00         6
     show_hn       0.94      1.00      0.97      4903
       story       1.00      1.00      1.00    126852

    accuracy                           1.00    137215
   macro avg       0.72      0.75      0.74    137215
weighted avg       1.00      1.00      1.00    137215

Confusion Matrix
[[  5454      0      0      0]
 [     0      0      0      6]
 [     0      0   4903      0]
 [   282      0    293 126277]]
Per class wise accuracy:
[1.         0.         1.         0.99546716]
Overall Accuracy: 99.57657690485733
```

Configuration:

- Delta = 0.5
- Stop-words = true

Observations:

- Here we can see that in the confusion matrix, the misclassification is reduced as we removed stop-words.
- As stop-word does not help us much in classification, so when we removed the stop-words, it leads to an increase in the accuracy of the classifier.
- Removal of the stop-words greatly impacted on the recall and f1-score of class ask_hn and show_hn. Hence, the increase in precision, recall, and f1-score leads to an increase in per class accuracies.

5.3 Word Length Filtering:

```
                precision    recall  f1-score   support

       ask_hn       0.20      1.00      0.33      5454
         poll       0.00      0.33      0.00         6
      show_hn       0.51      1.00      0.67      4903
        story       1.00      0.71      0.83    126852

     accuracy                           0.73    137215
    macro avg       0.43      0.76      0.46    137215
weighted avg       0.95      0.73      0.80    137215

Confusion Matrix
[[ 5454     0     0     0]
 [    0     2     0     4]
 [    0     0  4903     0]
 [22371 10275  4771 89435]]
Per class wise accuracy:
[1.         0.33333333 1.         0.70503421]
Overall Accuracy: 72.72820026964982
```

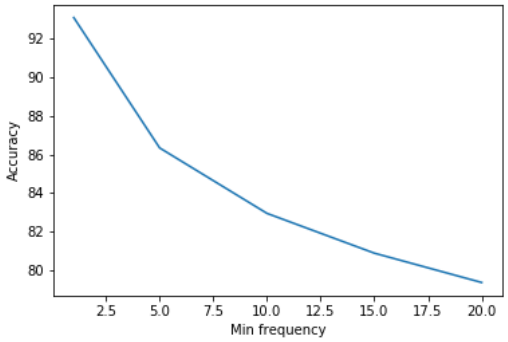
Configuration:

- Delta = 0.5
- Word Length Filtering = true

Observations:

- Here we can see that the accuracy is falling with word length filtering.
- One of the reasons for explaining this behavior can be removal of informative words from vocab resulting in underfitting of the model.

5.4 Infrequent Word Filtering:
Least frequent words:



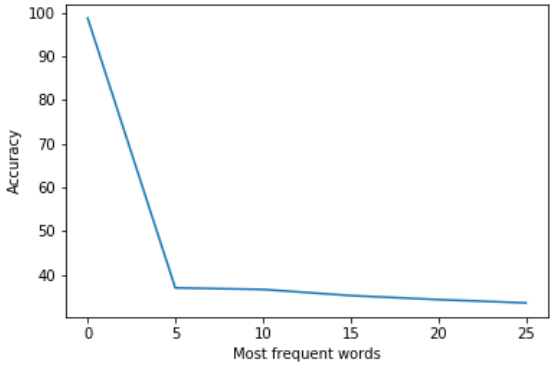
Configuration:

- Delta=0.5
- Remove words with frequencies :1,5,10,15,20

Observation:

- In this, we can see that on given data set our accuracy is decreasing with gradually removing the words with least frequencies.
- This is happening because our data set contains the least frequent number in classes such as poll, ask_hn, and show_hn. While it is true that they are also present in story class but removing words from other three classes will hit the accuracy and will be more biased toward story.

Most frequent words:



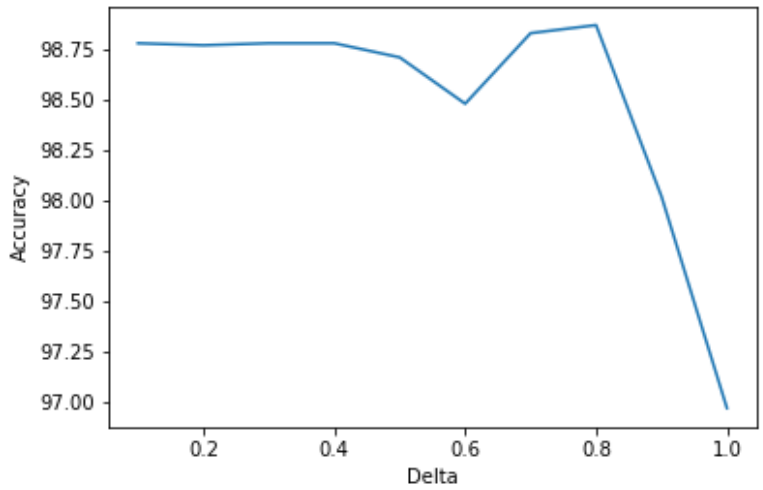
Configuration:

- Delta=0.5
- Remove top percentage=5%,10%,15%,20%,25%

Observation:

- Accuracy is falling at a great rate first then it is decreasing little by little.
- This tread is due to fact that it will greatly affect story class and will cause misclassification.

5.5 Smoothing:



Configuration:

- Delta: From 0.1 to 1.0
- Whole data set

Observations:

- On gradually improving delta from 0.1 to 1.0, we found that there is a decrease of 1 in overall accuracy.

- The overall decrease in accuracy is due to an increase in bias caused by larger smoothening values. Too much distribution is given to non-existing values resulting in decrease in efficiency of the classifier.

5.6 Overall Observations of Experiments:

- With a lower value of delta more accurate value is being produced.
- Word length filtering is having a negative effect on accuracy as important words can be removed from vocabulary.
- Removing stop words is improving accuracy because of the fact they are common in language and does not contain much information during classification.
- On removing most frequent words the accuracy decreases drastically as you are removing most informative words from the vocabulary.
- On removing the least frequent words our accuracy is decreasing but not like above. In this data set it is decreasing due to the fact this data set is more biased toward story.

6 Challenges Faced and Solutions:

During the project, we faced many challenges regarding efficiency and building vocabulary.

1. Efficiently reading data: We used pandas and CSV rstrip method to lead data into our program. While rstrip was the fastest one, we opted for pandas to do the task as it was the safer option.
2. Handling bigram: To minimize the complexity we focused on building bigram as a noun-noun.
3. Testing Model: It was too slow to load model and test data with the help of the data frame, so we switched to a dictionary to make this fast.

7 Future Work:

- Introduction of Author name while classifying the testing data. It can also give us an idea of what to predict when there is a situation such as ask_hn and story have equal probabilities and we need a tie-breaker.
- Use n-gram like trigram and more to analyze the performance of classification.
- Employ stemming and see how accuracy changes.
- Implement sentiment analysis on the Title and analyze whether there is any relation between Post Type and Sentiments.
- TF IDF model gives priority to words that can classify class more precisely.

8 Why this project is important in real life?

In today's world, people spend most of there time online and during that time they interact with the posts. This thing is very important for knowing the interest of people. From this, companies can analyze the data collected and then can analyze to sell their products.

9 Contributions:

TASK	NAME
Reading and analyzing data	Nishant Saini
Data cleaning	Dhwani Sondhi
Building model	Nishant Saini and Dhwani Sondhi
ML classifier	Dhwani Sondhi
Implementing sklearn classifier	Nishant Saini
Experiments	Nishant Saini and Dhwani Sondhi

10 References:

1. https://jupyter.org/
2. https://pandas.pydata.org/
3. https://www.nltk.org/
4. https://matplotlib.org/
5. https://scikit-learn.org/stable/
6. https://numpy.org/