Deep Learning Midterm Report

Jae Goan Park
Jin Yong Shin

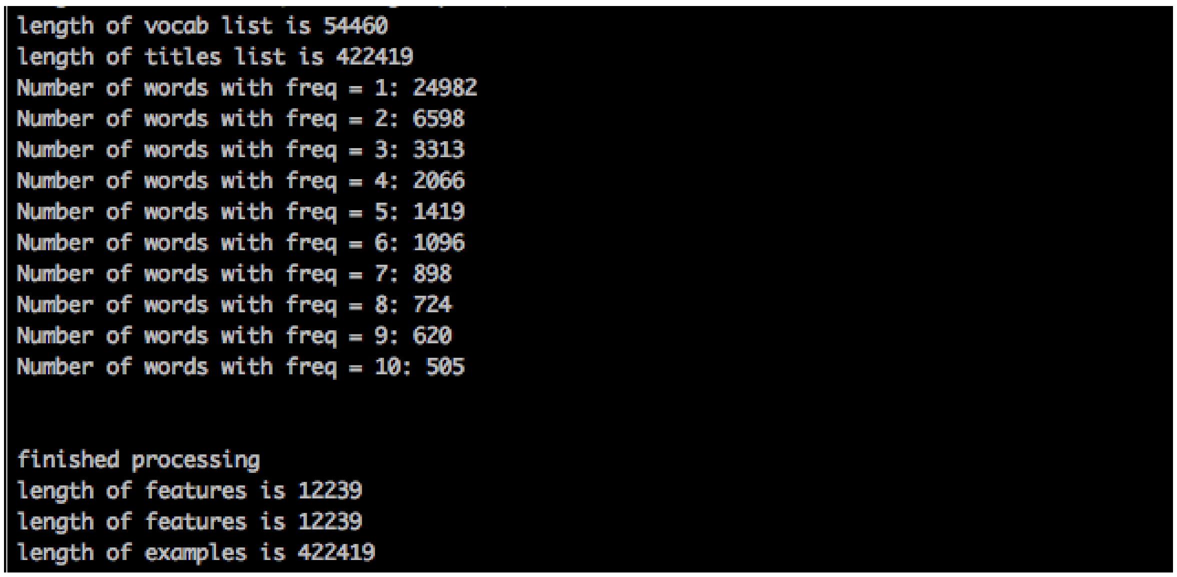
Github Address: https://github.com/jaegoan9/News-Classification-Project

Coding Progress

So far, we have made most of the progress on the machine learning aspect of the news article classification project. Data extraction and feature extraction were not as easy as we had thought. The News Title corpus database we got from Kaggle contained nearly 420,000 examples of news article titles and trying to run a machine learning model on a set of features derived from the bag of words model resulted in a disaster every time. To employ the bag of words model for this project, we created a unique set of vocabulary words from every single example in the dataset, and this initially consisted of around 55000 words. Some methods used for creating this set of vocabulary includes, removing punctuation, numbers, and stemming every single word to have words in different tenses combined into one word for an accurate representation of vocabulary in all of the examples. However, when we tried to create a feature list (list of lists of features for every single example), this ended up terminating the program midway during feature extraction since each list of features contained 55000 data points (frequency of each word per sentence), and this meant that we were doing 55,000 x 420,000 operations just to create the full set of features.

Resolving the issue with optimization

Having realized that we needed to optimize the feature extraction process, we decided to first reduce the set of vocabulary words by changing the methodology for word frequency count and limiting the vocab list to more relevant (frequent) words. Instead of measuring unique word frequency within each example of the dataset, we measured the frequency of unique words across the entire set of examples and filtered out words with low frequencies (such as 1 or 2). By implementing this groundbreaking change, we were able to reduce the full common set of words to around 12000 words, which meant we reduced the total number of operations necessary down around a factor of 4.5 (which is pretty big in our case).



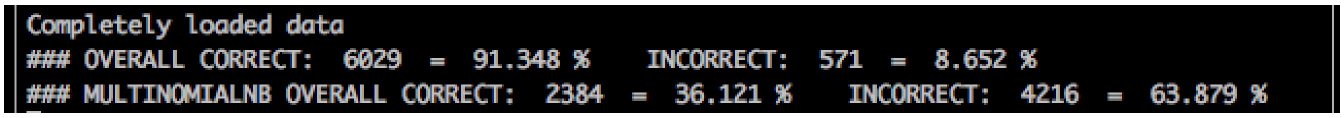
```
length of vocab list is 54460
length of titles list is 422419
Number of words with freq = 1: 24982
Number of words with freq = 2: 6598
Number of words with freq = 3: 3313
Number of words with freq = 4: 2066
Number of words with freq = 5: 1419
Number of words with freq = 6: 1096
Number of words with freq = 7: 898
Number of words with freq = 8: 724
Number of words with freq = 9: 620
Number of words with freq = 10: 505


finished processing
length of features is 12239
length of features is 12239
length of examples is 422419
```

However, we noticed that this still did take quite a bit of time despite applying fairly strict and efficient optimization techniques. Instead of further optimizing the model, we decided to change how we run the actual program. Instead of creating the full set of feature vectors all at once, we instead outputted feature vectors, segment by segment and dumped them into json files (1000 per file). This was very helpful in optimizing the program as we did not have to re-create feature vectors every single time and were able to quickly read in data from the json files.

Machine Learning Component

To evaluate our bag of words model with machine learning algorithms, we used a Decision Tree Classifier, Support Vector Machine, and Multinomial Naïve Bayes classifier. We obtained around 92% accuracy with the Decision Tree Classifier (labeled "OVERALL CORRECT" in bottom screenshot", 36% accuracy with the Multinomial Naïve Bayes Classifier. The results for the SVM model were inconclusive, as we have not found a way to optimize fitting the support vector machine to make in run in a reasonable amount of time.



```
Completely loaded data
### OVERALL CORRECT:  6029  =  91.348 %    INCORRECT:  571  =  8.652 %
### MULTINOMIALNB OVERALL CORRECT:  2384  =  36.121 %    INCORRECT:  4216  =  63.879 %
```

<u>Next Step</u>

We were slightly surprised by the performance of the multinomial naïve bayes classifier, since multinomial naïve bayes usually works fairly well for "classification with discrete features (e.g. word counts for text classification)". We are still investigating whether we should try to improve this model for better accuracy or if we should just search for different models.

Once we are done with the machine learning component of this project, we will move onto deep learning and its applications. Stay tuned for the final report!