CSCI6380 Data Mining: Assignment #2

Adnan Kivanc Corut

University of Georgia — September 24, 2020

Naive Bayes Text Classifier

1 Introduction

In this assignment, I try to create text classifiers using Naive Bayes model to identify the author of a text. To do this, I will be using texts from 7 different authors. Training data consists 22 different text files, and test data consists 7 different text files. I will be using both Multinominial and Bernoulli Naive Bayes models for the classification. Lastly, I will be testing different parameters, data processing and models to improve the accuracy of the model. Below I first explain my approach such as how the data were processed, which parameters and models were used. Then, I will present the accuracy results from different tests.

2 Approach

2.1 Data Processing

I first read the text files and lowercase all the words. Then, all words in a given text file were tokenized using `word_tokenize()` from nltk.tokenize package. Next, I removed stop words, punctuation, and digits from the list of tokens. After this cleaning step, stemming was performed to reduce tokens to their root form (for example, connection to connect) using `SnowballStemmer()` from nltk.stem package. After cleaning and stemming, tokens were splitted into 500 chunks using python list comprehension and store each 500 chunk in a row of numpy array. After breaking the text file into chunks of 500 tokens, tokens in each row of the array converted into string using `map()` method. Last, by iterating through each text file, corpus data was created by storing each text in rows of pandas dataframe as chunks of 500 tokens. First column of the corpus data consists of words (500 tokens in each row as string) and the second column is the corresponding author of the text. Using this approach, 22 test text files were converted into corpus data and the first column (words/text) was designated as training data and the second column (author names) was designated as training target. Same was done to 7 test files. This corpus data was later vectorized and used to feed text classifiers. Vectorization was done by using `CountVectorizer()` of scikit-learn.

2.2 Experiments

I used two different Naive Bayes model and tested different parameters to try to improve accuracy of the model. Table 1 shows different parameters that were used to test improvement of the accuracy of the

	n(tokens)	Cleaning	Stemming	TF-IDF	fitPrior
Experiment 1	500	True	True	+/-	False/True
Experiment 2	500	False	False	+/-	False/True
Experiment 3	500	False	True	+/-	False/True
Experiment 4	50	False	False	+/-	False/True
Experiment 5	250	False	False	+/-	False/True
Experiment 6	1000	False	False	+/-	False/True
Experiment 7	5000	False	False	+/-	False/True

Table 1: Parameters used to test accuracy.

model. I tested different number of tokens when breakin each text into "n" numbers of tokens. The effect of removing stop words, punctuation, and digits and the effect of stemming were also tested. In each experiment, we run the model with and without using TF-IDF (Term Frequency times inverse document frequency) which normalizes the weightage of common words. I also tested parameter fitPrior as False and True in each experiment. fitPrior parameter decides whether the model to learn class prior probabilities or not. I run each experiment separately with both Multinominial and Bernoulli Naive Bayes models.

3 Results

Table 2 shows the accuracies of each experiments with using both Multinominial and Bernoulli Naive Bayes models. Overall, these experiments were not so effective to improve the accuracy results I first obtained with default settings. Generally, Multinominial Naive Bayes performed better than Bernoulli Naive Bayes. The resulsts showed that, removing stop words, punctuation, and digits caused a decrease in accuracy. While using smaller numbers of tokens (such as n=50) to break each text results in lower accuracy, increased numbers of "n" did not affect the accuracy dramatically. Overall, setting `fitPrior` parameter to false seems to have positive effect on accuracy. I also observed that while using TF-IDF had no effect on Bernoulli Naive Bayes model, it decreases the accuracy when used with Multinominial Model. Lastly, the best accuracy result, 0.756595, was obtained in experiment 2 with `fitPrior=False`. To improve accuracy further, other methods such as Part-of-Speech(POS) tagging can be used. Additionally, different tokenizers or different algorithms, such as Support Vector Machines, can be implemented.

Experiment 1	**MultinomialNB**	**BernoulliNB**
Default Settings (n=500, clean=True, stem=True)	0.738444	0.691093
Default Settings with TF-IDF	0.603156	0.691093
Default Settings with fitPrior=False	0.740698	0.694475
Experiment 2	**MultinomialNB**	**BernoulliNB**
No Cleanning and Stemming (n=500, clean=False, stem=False)	0.755102	0.690890
No Cleanning and Stemming with TF-IDF	0.465903	0.690890
No Cleanning and Stemming with fitPrior=False	0.756595	0.695868
Experiment 3	**MultinomialNB**	**BernoulliNB**
Stemming but No Cleaning (n=500, clean=False, stem=True)	0.750845	0.739571
Stemming but No Cleaning with TF-IDF	0.625704	0.739571
Stemming but No Cleaning with fitPrior=False	0.750845	0.739571
Experiment 4	**MultinomialNB**	**BernoulliNB**
No Cleanning and Stemming (n=50, clean=False, stem=False)	0.651811	0.632620
No Cleanning and Stemming (n=50) with TF-IDF	0.504910	0.632620
No Cleanning and Stemming (n=50) with fitPrior=False	0.664523	0.647425
Experiment 5	**MultinomialNB**	**BernoulliNB**
No Cleanning and Stemming (n=250, clean=False, stem=False)	0.746948	0.69464
No Cleanning and Stemming (n=250) with TF-IDF	0.519053	0.69464
No Cleanning and Stemming (n=250) with fitPrior=False	0.750186	0.70112
Experiment 6	**MultinomialNB**	**BernoulliNB**
No Cleanning and Stemming (n=1000, clean=False, stem=False)	0.751491	0.671968
No Cleanning and Stemming (n=1000) with TF-IDF	0.394632	0.671968
No Cleanning and Stemming (n=1000) with fitPrior=False	0.751491	0.671968
Experiment 7	**MultinomialNB**	**BernoulliNB**
No Cleanning and Stemming (n=5000, clean=False, stem=False)	0.733990	0.630541
No Cleanning and Stemming (n=5000) with TF-IDF	0.211822	0.630541
No Cleanning and Stemming (n=5000) with fitPrior=False	0.738916	0.630541

Table 2: Model accuracy results from different experiments using different parameters.