TEXT CLASSIFICATION

Karahan Sarıtaş - 2018400174
Department of Computer Engineering, Boğaziçi University
karahan.saritas@boun.edu.tr

ABSTRACT

In this assignment, we are expected to implement the multinomial Naive Bayes (NB) and the multivariate Bernoulli Naive Bayes algorithms for text classification. We will use the Reuters-21578 data set (the same data set from the first assignment). Reuters-21578 contains 21578 news stories from Reuters newswire. There are 22 SGML files, each containing 1000 news articles, except the last file, which contains 578 articles. There are a total of 118 topics (classes) and each article is classified into one or more topics.

1 Data Preprocessing

Firstly, we start our data preprocessing procedure by parsing the news from the Reuters dataset. We extract **<TITLE>**, **<BODY>**, **<LEWISSPLIT>** and **<TOPIC>** tags of each news. *NEWID* fields are used as document IDs. A set of regex patterns are used to extract relevant information. Documents are stored in a dictionary where *key* corresponds to document IDs and *value* corresponds to smaller dictionaries composed of a title, body, lewis split, and topic for each text. We ignore the texts without titles or without body components. After collecting all the textual information, we feed these documents into the tokenization pipeline. Our pipeline roughly consists of five steps: HTML unescaping, punctuation removal, digit removal, splitting, stopword removal, case-folding and stemming.

In HTML, escaping involves substituting certain special characters with alternative ones, typically <, >, ",$, and &. These characters hold distinct purposes within HTML documents. The fundamental idea of escaping is to encrypt the characters < and > and apostrophes to make them less identifiable as tags. Escaping HTML has a variety of uses, the most obvious of which is that it can be inserted into an HTML document without rendering to show code. It can be observed that our dataset contains examples of HTML escaping. We have to eliminate escaping characters before proceeding with tokenization. For this purpose, the built-in html (HyperText Markup Language support) library can be used in Python.

After removing the escape characters, we can remove the punctuation and digits. To remove the punctuation, we simply used the list in string.punctuation provided within the standard library. Upon removal of punctuation and digits, we split the text into tokens by whitespaces. This operation changes our data structure from *string* to *list*. Now we have a list of tokens in our hands.

Now, we apply *case-folding* to each token in our list, by reducing all letters to lowercase. Case-folding helps us to match instances of different words at the beginning of a sentence with the query of those words. On the other hand, such case folding can equate words that might better be kept apart (such as *Fed* and *fed*) [1, p. 30]. Case-folding doesn't reduce the number of tokens we have - but it is very likely to reduce the number of *unique* tokens in the corpus. After case-folding, we remove the stopwords from the list using a list of stopwords prepared beforehand.

Lastly, we apply *stemming* to each token in the list. We used a slightly modified version of the Porter Stemming algorithm, which has proved to be one of the most effective stemming algorithms empirically, provided here [2].The goal of both stemming is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form [1, p. 32]. After the stemming procedure, the number of unique tokens decreases since some of the different tokens will be reduced to the same root.

At the end of the data preprocessing, we had 31,706 unique words in the vocabulary.

2 Top 10 Topics

Top 10 topics: ['earn', 'acq', 'money-fx', 'grain', 'crude', 'trade', 'interest', 'ship', 'wheat', 'corn']
Number of documents labeled with more than one of the top 10 classes: 1793
Here, you can find the number of documents per topic both for training and test sets:

```
earn:
     training: 2709
     test: 1044
acq:
     training: 1488
     test: 643
money-fx:
     training: 460
     test: 141
grain:
     training: 394
     test: 134
crude:
     training: 349
     test: 161
trade:
     training: 337
     test: 113
interest:
     training: 289
     test: 100
ship:
     training: 191
     test: 85
wheat:
     training: 198
     test: 66
corn:
     training: 160
     test: 48
```

Vocab size for training set: 16544
Vocab size for test + training set: 19673
Vocab size for testset + training set + not-used: 20121
Number of documents labeled with more than one of the top 10 classes: 770

3 Parameter Tuning

I've implemented a training script so that it accepts multiple alpha values for hyperparameter tuning. Therefore, one can easily test out different alpha values on Multinomial NB/Bernoulli NB using the commands below:

```
$ python dataprocessor.py
$ python splitter.py
$ python train.py --model m --alpha 0.5 1.0 2.0
$ python test.py --model m --alpha X
```

where X is the alpha value with the best score on the development set. There are a total of 6135 documents in the training set, and 1285 of them are separated for the development set. While creating the development set, I iterated through the topics in the list. For each topic, I collected the documents in the whole dataset and split them into training and test sets according to the `lewis-splits`. Then, I extracted part of my training set (for each topic) as a development set for tuning my classifiers. A split ratio can be provided via the command line - if not, it defaults to 0.8.

4 Evaluation Results

For the Multinomial NB, the best alpha value turned out to be 0.5 among the values 0.1, 0.5, and 2. Here are the results from both the development set and the test set:

```
Multinomial Naive Bayes on Development Set
Alpha: 0.1
Correct predictions: 1198 out of: 1288
 -- micro-precision: 0.93  -- macro-precision: 0.905
 -- micro-recall: 0.778  -- macro-recall: 0.719
 -- micro-f1: 0.848  -- macro-f1: 0.777
>>> Evaluation is completed. Elapsed time: 0.454 seconds
--------------------------------------------------

Multinomial Naive Bayes on Test Set
Correct predictions: 2171 out of: 2315
 -- micro-precision: 0.938  -- macro-precision: 0.859
 -- micro-recall: 0.857  -- macro-recall: 0.683
 -- micro-f1: 0.895  -- macro-f1: 0.749
>>> Testing is completed. Elapsed time: 0.572 seconds
```

For the Bernoulli NB, the best alpha value turned out to be 0.5 among the values 0.1, 0.5 and 2. Here are the results from both the development set and the test set:

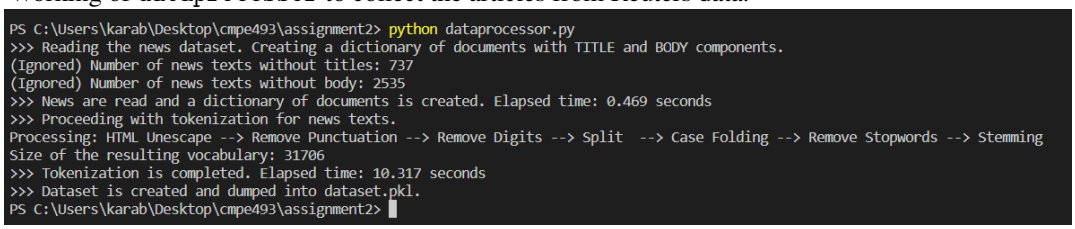
```
Bernoulli Naive Bayes on Development Set
Alpha: 0.01
Correct predictions: 1122 out of: 1288
 -- micro-precision: 0.871  -- macro-precision: 0.853
 -- micro-recall: 0.729  -- macro-recall: 0.673
 -- micro-f1: 0.794  -- macro-f1: 0.732
>>> Evaluation is completed. Elapsed time: 77.084 seconds
--------------------------------------------------

Bernoulli Naive Bayes on Test Set
Correct predictions: 2054 out of: 2315
 -- micro-precision: 0.811  -- macro-precision: 0.619
 -- micro-recall: 0.811     -- macro-recall: 0.614
 -- micro-f1: 0.811         -- macro-f1: 0.614
>>> Testing is completed. Elapsed time: 342.548 seconds
```

Based on the results of my randomization test, I obtained a p value that is less than 0.05. Consequently, I can conclude that there is a *statistically significant* difference between the two models and that Multinomial NB performs better than Bernoulli NB.

5 Screenshots

- Working of `dataprocessor` to collect the articles from Reuters data.



- Working of `splitter` to split the data into training and test sets.

```
PS C:\Users\karab\Desktop\cmpe493\assignment2> python splitter.py
Dataset is to be splitted into training, validation and test sets.
Top 10 topics:  ['earn', 'acq', 'money-fx', 'grain', 'crude', 'trade', 'interest', 'ship', 'wheat', 'corn']
Vocab size for training set: 16544
Vocab size for test + training set: 19673
Vocab size for testset + training set + not-used: 20121
Number of documents labeled with more than one of the top 10 classes: 770
earn:
     training: 2709
     test: 1044
acq:
     training: 1488
     test: 643
money-fx:
     training: 460
     test: 141
grain:
     training: 394
     test: 134
crude:
     training: 349
     test: 161
trade:
     training: 337
     test: 113
interest:
     training: 289
     test: 100
ship:
     training: 191
     test: 85
wheat:
     training: 198
     test: 66
corn:
     training: 160
     test: 48
Total number of documents in training: 6135 (development: 1285)
Total number of documents in test: 2315
Splits are created and dumped into splits.pkl.
PS C:\Users\karab\Desktop\cmpe493\assignment2>
```

- Working of `train` script to train the models and evaluate them on the development set.

(i)

```
PS C:\Users\karab\Desktop\cmpe493\assignment2> python train.py -m m --alpha 0.5
<---------------------------------------->
Multinomial Naive Bayes
>>> Training with the given alpha value(s) on training set. Evaluating the model on development set.
--------------------------------------------------
Alpha: 0.5
Correct predictions: 1196 out of: 1285
 -- micro-precision: 0.931  -- macro-precision: 0.927
 -- micro-recall: 0.767  -- macro-recall: 0.615
 -- micro-f1: 0.841  -- macro-f1: 0.68
>>> Evaluation is completed. Elapsed time: 0.405 seconds
PS C:\Users\karab\Desktop\cmpe493\assignment2>
```

(ii)

```
PS C:\Users\karab\Desktop\cmpe493\assignment2> python test.py -m m --alpha 0.5
<---------------------------------------->
Multinomial Naive Bayes
>>> Testing with the given alpha value on test set. Both training and development tests are used for training.
--------------------------------------------------
Correct predictions: 2198 out of: 2315
 -- micro-precision: 0.949  -- macro-precision: 0.928
 -- micro-recall: 0.867  -- macro-recall: 0.642
 -- micro-f1: 0.907  -- macro-f1: 0.682
>>> Testing is completed. Elapsed time: 0.59 seconds
PS C:\Users\karab\Desktop\cmpe493\assignment2>
```

- Working of `test` script to train the model on training + development set and test it on test set.

(i)

```
PS C:\Users\karab\Desktop\cmpe493\assignment2> python train.py -m b --alpha 0.5
<---------------------------------------->
Bernoulli Naive Bayes
>>> Training with the given alpha value(s) on training set. Evaluating the model on development set.
--------------------------------------------------
Alpha: 0.5
Correct predictions: 1080 out of: 1285
 -- micro-precision: 0.84  -- macro-precision: 0.85
 -- micro-recall: 0.693  -- macro-recall: 0.528
 -- micro-f1: 0.759  -- macro-f1: 0.592
>>> Evaluation is completed. Elapsed time: 78.415 seconds
PS C:\Users\karab\Desktop\cmpe493\assignment2>
```

(ii)

```
PS C:\Users\karab\Desktop\cmpe493\assignment2> python test.py -m b --alpha 0.5
<---------------------------------->
Bernoulli Naive Bayes
>>> Testing with the given alpha value on test set. Both training and development tests are used for training.
--------------------------------------------------
Correct predictions: 2016 out of: 2315
 -- micro-precision: 0.871  -- macro-precision: 0.671
 -- micro-recall: 0.796  -- macro-recall: 0.541
 -- micro-f1: 0.832  -- macro-f1: 0.586
>>> Testing is completed. Elapsed time: 150.065 seconds
PS C:\Users\karab\Desktop\cmpe493\assignment2>
```

- Working of `randomization_test` script to apply randomization test on our models.

```
PS C:\Users\karab\Desktop\cmpe493\assignment2> python randomization_test.py -ma 0.5 -ba 0.5
>>> Predictions of Multinomial NB are calculated.
Predictions of Multinomial NB are calculated. Elapsed time: 0.389 seconds
>>> Predictions of Bernoulli NB are calculated.
Predictions of Bernoulli NB are calculated. Elapsed time: 164.557 seconds
>>> Randomization test is started.
Randomization test is completed. Elapsed time: 3.632 seconds
>>> p-value: 0.010989
>>> The difference between the two models is statistically significant.
```

References

[1] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval*. Cambridge University Press, 2008.

[2] Martin Porter. Python implementation of the porter stemming algorithm, 2002.