University of Macau

CISC3025 – Natural Language Processing

Project#2, 2023/2024

(Due date: **18th March 2024**)

Problem Description

This project asks you to implement a naïve-Bayes text classification model and evaluate the performance of the implemented model in the Reuters dataset. The bag-of-word model will be used as the representation of the documents.

The *bag-of-word* model represents a text as the bag of its words, disregarding grammar and word order but keeping the word frequency in the document. Due to the variety of the words in the real world, using all words in the dataset will cause high demand for memory space and computing resources. So feature selection will be assigned to select useful words for the text-classification task. The selected words, called features, will be used for the machine learning model. The simplest feature selection method considers the word frequency. It simply ignores words whose word frequency is less than a threshold.

The *naïve-Bayes* text classification model uses the Bayes rule to determine the class distribution possibility of the documents. For a document d and a class c:

$$P(c|d) = \frac{P(d|c)p(c)}{P(d)}$$

In real-world classification tasks, the model will assign the document with the class label which has the maximum probability:

$$\begin{aligned} c_{MAP} &= \underset{c \in C}{\operatorname{argmax}} P(c|d) \\ &= \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)p(c)}{P(d)} \\ &= \underset{c \in C}{\operatorname{argmax}} P(d|c)p(c) \end{aligned}$$

where $P(d|c)$ is the posterior probability, and $p(c)$ is the prior probability. As the bag-or-words model is used as the representation of the documents, the $P(d|c)$ can be represented as $P(x_1, x_2, \dots, x_n|c)$. Hence, the above equation can be expressed as:

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \dots, x_n|c)p(c)$$

In *naïve Bayes* model, it assumes the feature probabilities are independent given the class c, and is defined as:

$$P(x_1, x_2, \dots, x_n|c) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c)$$

In sum, the *naïve Bayes* classifier uses the posterior probability of the word and the prior

probability of the class to assess the class attribution possibility of the documents.

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{x \in X} P(x|c)$$

To estimate the class probability, we can simply count the number of documents belonging to a category j:

$$\hat{p}(c_j) = \frac{DocCount(C = c_j)}{N_{doc}}$$

For those of the posterior probabilities of words, we use the following equation for the computation:

$$\hat{p}(w_i|c_j) = \frac{Count(w_i, c_j)}{\sum_{w \in V} Count(w, c_j)}$$

To avoid the zero-probability, we use the Laplace smoothing for *naïve Bayes*:

$$\hat{p}(w_i|c_j) = \frac{Count(w_i, c_j) + 1}{(\sum_{w \in V} Count(w, c_j)) + |V|}$$

To evaluate the performance of the classification model, we consider *Precision*, *Recall* and the F_1 *score*. We divide the sample prediction situation into four categories, as shown in Table 1.

	True Condition	
Predicted Condition	True Positive (TP)	False Positive (FP)
	False Negative (FN)	True Negative (TN)

Table 1 Categories for prediction

The Precision (P) stands for the quality of the samples which are predicted to be true:

$$P = \frac{TP}{TP + FP}$$

The Recall (R) concentrates on how many positive samples were successfully identified:

$$R = \frac{TP}{TP + FN}$$

The F_1-Score combines these two indicators:

$$F_1 = 2 * \frac{P * R}{P + R}$$

Training and Test Data

The training and test data for this project come from the Reuters Corpus, which contains 10,788 news documents totaling 1.3 million words. The documents have been classified into 90 topics, and grouped into two sets, called "*training*" and "*test*".

For simplicity, the data has been pre-processed. In this simplified dataset, there are only 5 categories in total: *'crude'*, *'grain'*, *'money-fx'*, *'acq'* and *'earn'*. The training set is stored in the *train.json*, and the test set is stored in the *test.json* file, which contains 5,787 and 2,298 text samples, respectively. Each text sample belongs to one topic.

In the train.json and the test.json file, each record represents a document, consisting of the file *id*, *category*, and *raw* text.

Requirements

1. Write a Python program to count the word in the corpus. The output of the program should be a text file, named 'word_count.txt'. The first line contains the total document frequency in five classes, in the order of: 'crude', 'grain', 'money-fx', 'acq' and 'earn', separated by a space. In the following lines, each line contains a word and its frequency in five classes, in the same order of the classes, separated by space.

```
Word_count.txt:
2300 1400 768 520 123
apple 5 10 7 8 6
banana 10 9 16 24 6
candy 102 0 400 450 200
silly 30 20 15 23 64
the 350 120 260 730 110
.
.
.
```

2. Write a program to choose the most frequent 10, 000 words as the feature words, and use the frequency obtained in requirement 1 to calculate the total word frequency in each class. **Notice that only words recognized as features are taken into consideration.** The output of the program should be a text file, named 'word_dict.txt', which contains the information about the feature words. The first line contains the total feature word frequency in five classes, in order of: 'crude', 'grain', 'money-fx', 'acq', and 'earn', separated by a space. In the following lines, each line contains a word and its frequency in five classes, in the same order as above, separated by space.

3. Use the above formula to calculate the posterior probability of each feature word and the prior probability of the class. The output of your program should be a text file, named 'word_probability.txt', The first line contains the prior probability of the five classes, in the order of: 'crude', 'grain', 'money-fx', 'acq', and 'earn', separated by space. In the following lines, each line contains a word and its posterior probability in five classes, separated by space.

4. Implement the *naïve Bayes* classifier to assign class labels to the documents in the test set, and store the file_id and class label to 'classification_result.txt'. **Noted that each document should be assigned a class label.** In this file, each line contains a file id and its class label, separated by a space.

5. Use the F_1 score to assess the performance of the implemented classification model.

6. (Optional) Add more features to the classification model and report the F_1 score of the new model with your enhanced features.

7. Write a report to introduce your work about the assignment.

The Starter Code

You may find the starter code 'naïve-bayes.py' under the 'Project2' directory. The following command will preprocess the data and generate the required training data.

```
$python naive-bayes.py -pps train.json train.preprocessed.json
```

The following command produces the count file for the words:

```
$python naive-bayes.py -cw train.preprocessed.json word_count.txt
```

You may select the top 1000 frequent words as features for constructing your NB classifier:

```
$python naive-bayes.py -fs word_count.txt 10000 word_dict.txt
```

The word probabilities are computed using the following command:

```
$python naive-bayes.py -cp word_count.txt word_dict.txt
word_probability.txt
```

Moreover, you can use the following command to classify the samples:

```
$python naive-bayes.py -cl word_probability.txt test.preprocessed.json
classification_result.txt
```

Finally, the F_1 score can be calculated as:

```
$python naive-bayes.py -f1 'test.json' 'classification_result.txt'
```

Submissions

You need to submit the following materials:

- **Runnable source code**
  - You need to make sure your code functions well according to the project requirements.
  - Of course, any other optimization designs are allowable. You can describe your genius designs in your report.
- **The output files**
  - The output files, 'word_count.txt', 'word_dict.txt', 'word_probability.txt', and 'classification_result.txt', produced by your program.
- **Report**
  - You need to submit a report to introduce your work. The report should contain the following information:
    - An introduction about your project, what is it about? What did you achieve?
    - Description of the project, what did you do? What is your approach, design, or method? What did/didn't work? How did you solve it? What did you achieve? What are the results?
    - Conclusions, what was accomplished/learned? What would you have done differently? What would you suggest for future work?