Megan Tan, A20527707

CS 585 Spring 2024 Programming Assignment #02

Due: **Sunday, March 24, 2023 at 11:59 PM CST**
Points: **100**

Instructions:

1. Place **all your deliverables (as described below) into a single ZIP** file named:

`LastName_FirstName_CS585_Programming02.zip`

2. Submit it to Blackboard Assignments section before the due date. **No late submissions will be accepted**.

Objectives:

1. (100 points) Implement and evaluate a Naïve Bayes classifier algorithm.

Task:

Your task is to implement, train, and test a Naïve Bayes classifier using a publicly available data set. **You can work in groups of two or by yourself. Two individual students / groups can use the same data set**.

Data set:

Pick a publicly available data (**follow the guidelines provided in Blackboard**) set first and do an initial exploratory data analysis.

Deliverables:

Your submission (if you are working as a group of two, both partners should submit the same work) should include:

- Python code file(s). Your py file should be named:

  `CS585_P02_AXXXXXXXX.py`

  where `AXXXXXXXX` is your IIT A number (this is REQUIRED!). If your solution uses multiple files, makes sure that the main (the one that will be run to solve the problem) is named that way and others include your IIT A number in their names as well.

- Presentation slides in PPTX or PDF format. Name it:

  `LastName_FirstName_CS585_P02_Slides.pptx or pdf`

- This document with your observations and conclusions. You should rename it to:

  `LastName_FirstName_CS585_P02.pdf`

Implementation:

Your task is to implement (**from scratch – you can't use out-of-the-box Python package classifier**), train, and test a Naïve Bayes classifier (as outlined in class) and apply it to classify sentences entered using keyboard.

Your program should:

- Accept one (1) command line argument, i.e. so your code could be executed with

  `python CS585_P02_AXXXXXXXX.py TRAIN_SIZE`

  where:

  - `CS585_P02_AXXXXXXXX.py` is your python code file name,
  - `TRAIN_SIZE` is a number between 20 and 80 defining the size (in percentages) of the training set. For example: `60` would mean **FIRST** (as ordered in the dataset file) 60% of samples. **Note that your test set is always going to be the LAST (as ordered in the dataset file) 20% of samples.**

  Example:

  `python CS585_P01_A11111111.py YES`

  If the number of arguments provided is NOT one (none, two or more) or the `TRAIN_SIZE` argument is out of the specified range, assume that the value for `TRAIN_SIZE` is `80`.

- Load and process input data set:
  - Apply any data clean-up / wrangling you consider necessary first (mention and discuss your choices in the Conclusions section below).
  - Text pre-processing:
    - treat every document in the data set as a single sentence, even if it is made of many (no segmentation needed),

- Train your classifier on your data set:
  - assume that vocabulary V is the set of ALL words in the data set,
  - divide your data set into:
    - training set: FIRST (as they appear in the data set) `TRAIN_SIZE` % of samples / documents,
    - test set: LAST 20 % of samples / documents,
  - use **binary** BAG OF WORDS with **"add-1" smoothing** representation for documents,
  - train your classifier (find its parameters. HINT: use Python dictionary to store them),

- Test your classifier:
  - use the test set to test your classifier,
  - calculate (and display on screen) following metrics:

- number of true positives,
- number of true negatives,
- number of false positives,
- number of false negatives,
- sensitivity (recall),
- specificity,
- precision,
- negative predictive value,
- accuracy,
- F-score,

- Ask the user for keyboard input (a single sentence S):
  - use your Naïve Bayes classifier to decide (HINT: use log-space calculations to avoid underflow – but bring it back to linear space after!) which class S belongs to,
  - display classifier decision along with P(CLASS_A |S) and P(CLASS_B | S) values on screen

Your program output should look like this (if pre-processing step is NOT ignored, output NONE):

```
Last Name, First Name, AXXXXXXXX solution:
Training set size: 80 %

Training classifier…
Testing classifier…
Test results / metrics:

Number of true positives: xxxx
Number of true negatives: xxxx
Number of false positives: xxxx
Number of false negatives: xxxx
Sensitivity (recall): xxxx
Specificity: xxxx
Precision: xxxx
Negative predictive value: xxxx
Accuracy: xxxx
F-score: xxxx

Enter your sentence:

Sentence S:

<entered sentence here>

was classified as <CLASS_LABEL here>.
P(<CLASS_A> | S) = xxxx
P(<CLASS_B> | S) = xxxx
```

```
Do you want to enter another sentence [Y/N]?
```

If user responds `Y`, classify new sentence (you should not be re-training your classifier).

where:

- `80` would be replaced by the value specified by `TRAIN_SIZE`,
- `xxxx` is an actual numerical result,
- `<entered sentence here>` is actual sentence entered y the user,
- `<CLASS_LABEL here>` is the class label decided by your classifier,
- `<CLASS_A>`, `<CLASS_B>` are available labels (SPAM/HAM, POSITIVE/NEGATIVE, etc.).

Classifier testing results:

Enter your classifier performance metrics below:

With `TRAIN_SIZE` set to 80:	With `TRAIN_SIZE` set to 20 (not 80)
Number of true positives: 1788 Number of true negatives: 5421 Number of false positives: 791 Number of false negatives: 529 Sensitivity (recall): 0.77169 Specificity: 0.87267 Precision: 0.69329 Negative predictive value: 0.91109 Accuracy: 0.84523 F-score: 0.73039	Number of true positives: 355 Number of true negatives: 5907 Number of false positives: 2224 Number of false negatives: 43 Sensitivity (recall): 0.89196 Specificity: 0.72648 Precision: 0.13765 Negative predictive value: 0.99277 Accuracy: 0.7342 F-score: 0.23849

What are your observations and conclusions? When did the algorithm perform better? a summary below

Summary / observations / conclusions
When train size = 80

```
Please enter your sentence: disneyland was awesome! I had the best time. It was fun and amazing
Sentence S:

     disneyland was awesome! I had the best time. It was fun and amazing

    was classified as POSITIVE.
    P(POSITIVE | S) = 1.607e-20
    P(NEGATIVE | S) = 5.5175e-24
Do you want to enter another sentence? [Y/N] y
Please enter your sentence: disneyland sucked! It was terrible. I felt awful the whole time
Sentence S:

     disneyland sucked! It was terrible. I felt awful the whole time

    was classified as NEGATIVE.
    P(POSITIVE | S) = 3.4438e-29
    P(NEGATIVE | S) = 6.795900000000001e-29
```

When TRAIN_SIZE = 20

```
Please enter your sentence: disneyland was awesome! I had the best time. It was fun and amazing
Sentence S:

     disneyland was awesome! I had the best time. It was fun and amazing

    was classified as POSITIVE.
    P(POSITIVE | S) = 6.72480000000000006e-21
    P(NEGATIVE | S) = 9.2875e-25
Do you want to enter another sentence? [Y/N] y
Please enter your sentence: disneyland sucked! It was terrible. I felt awful the whole time
Sentence S:

     disneyland sucked! It was terrible. I felt awful the whole time

    was classified as POSITIVE.
    P(POSITIVE | S) = 5.4613e-30
    P(NEGATIVE | S) = 2.3957000000000003e-30
```

A larger training set leads to much better results as per the reported metrics. When inputting user sentences, the model with the larger training set also more accurately classifies the user sentences (shown above).

However, the model is still highly inaccurate as seen in the example below.

```
     disney sucked

    was classified as POSITIVE.
    P(POSITIVE | S) = 4.968894897491188e-08
    P(NEGATIVE | S) = 2.2667890287655082e-08
Do you want to enter another sentence? [Y/N] y
Please enter your sentence: sucks
Sentence S:

     sucks

    was classified as POSITIVE.
    P(POSITIVE | S) = 9.411756646693865e-06
    P(NEGATIVE | S) = 6.312675167725326e-06
```

This could be due to a much larger number of positive sentiments in the data with nearly 80% of the review being positive. Downsampling the training set can help make the model more accurate. K fold validation can also make the model more accurate.