Xiaonan Peng and Junhui Zhan

**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

1. 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 \_\_\_ (not 80) |
| Number of true positives: 56892  Number of true negatives: 12162  Number of false positives: 4788  Number of false negatives: 4874  Sensitivity (recall): 0.9211  Specificity: 0.7175  Precision: 0.9224  Negative predictive value: 0.7139  Accuracy: 0.8773  F-score: 0.9217 | Number of true positives: 57007  Number of true negatives: 12044  Number of false positives: 4906  Number of false negatives: 4759  Sensitivity (recall): 0.9230  Specificity: 0.7106  Precision: 0.9208  Negative predictive value: 0.7168  Accuracy: 0.8772  F-score: 0.9219 |

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

|  |
| --- |
| **Summary / observations / conclusions** |
| The algorithm performance was influenced by various factors related with input data and text pre-processing:  **Binarization**  We found that applying Binary Naive Bayes does not improve the performance, especially it leads to much more false positives compared to multinomial Naive Bayes.   |  |  | | --- | --- | | **Multinominal Naive Bayes**  Number of true positives: 4302  Number of true negatives: 713  Number of false positives: 498  Number of false negatives: 223  Sensitivity (recall): 0.9507  Specificity: 0.5888  Precision: 0.8962  Negative predictive value: 0.7618  Accuracy: 0.8743  F-score: 0.9227  Time consumed in seconds: 12.3055s | **Binary Naive Bayes**  Number of true positives: 4525  Number of true negatives: 2  Number of false positives: 1209  Number of false negatives: 0  Sensitivity (recall): 1.0000  Specificity: 0.0017  Precision: 0.7892  Negative predictive value: 1.0000  Accuracy: 0.7892  score: 0.8822  Time consumed in seconds: 12.2179s |   **Stemmer**  We have tried the Poster Stemming Algorithm, but it does not help much in terms of precision or accuracy, except for doubling the execution time. So, instead, we use simplified stemming which just removing some postfixes like “ing” and “ed” from the words.   |  |  | | --- | --- | | **Porter Stemming**  Number of true positives: 4253  Number of true negatives: 747  Number of false positives: 464  Number of false negatives: 272  Sensitivity (recall): 0.9399  Specificity: 0.6168  Precision: 0.9016  Negative predictive value: 0.7331  Accuracy: 0.8717  F-score: 0.9204  Time consumed in seconds: 23.9632s | **Simplified stemming**  Number of true positives: 4302  Number of true negatives: 713  Number of false positives: 498  Number of false negatives: 223  Sensitivity (recall): 0.9507  Specificity: 0.5888  Precision: 0.8962  Negative predictive value: 0.7618  Accuracy: 0.8743  F-score: 0.9227  Time consumed in seconds: 12.7031s |   **Stop words**  Remove stop words can increase a little bit Sensitivity , Negative predictive value, accuracy and F-score, but decrease Specificity and Precision   |  |  | | --- | --- | | Keep stop words  Number of true positives: 4201  Number of true negatives: 767  Number of false positives: 444  Number of false negatives: 324  Sensitivity (recall): 0.9284  Specificity: 0.6334  Precision: 0.9044  Negative predictive value: 0.7030  Accuracy: 0.8661  F-score: 0.9162  Time consumed in seconds: 12.5605s | remove stop words  Number of true positives: 4302  Number of true negatives: 713  Number of false positives: 498  Number of false negatives: 223  Sensitivity (recall): 0.9507  Specificity: 0.5888  Precision: 0.8962  Negative predictive value: 0.7618  Accuracy: 0.8743  F-score: 0.9227  Time consumed in seconds: 12.3245s |   **1 char word**  Remove 1 char words does not help much.   |  |  | | --- | --- | | **Keep 1 char words**  Number of true positives: 4299  Number of true negatives: 717  Number of false positives: 494  Number of false negatives: 226  Sensitivity (recall): 0.9501  Specificity: 0.5921  Precision: 0.8969  Negative predictive value: 0.7603  Accuracy: 0.8745  F-score: 0.9227  Time consumed in seconds: 12.3169s | **Remove 1 char words**  Number of true positives: 4302  Number of true negatives: 713  Number of false positives: 498  Number of false negatives: 223  Sensitivity (recall): 0.9507  Specificity: 0.5888  Precision: 0.8962  Negative predictive value: 0.7618  Accuracy: 0.8743  F-score: 0.9227  Time consumed in seconds: 12.2915s |   **Negations**  Add prefix `not\_` to every word between negation and the following punctuation to adjust classifying sentence with negations.   |  |  | | --- | --- | | **Negations**  Enter your sentence:  Sentence S:  I don't like the book  was classified as bad  P(good|S)=0.1306  P(bad|S)=0.8694 | **No Negations**  Enter your sentence:  Sentence S:  I don't like the book  was classified as good  P(good|S)=0.7571  P(bad|S)=0.2429 |   **Further improvement**  We find that the trained classifier tends to classify sentences as “good” label. As more than 70% records are labeled as “good”. So according the the equation below, P(y=good) takes high weight.  ymap ∝ P(y)\*P(w1|y)\*...\*P(w2|y)  Take a example of classifying this sentence “I dislike it”. (Training based on 1000 records)   |  |  | | --- | --- | | P(y=good) = 0.76  P(dislike|good) = 2.02e-05  ... | P(y=bad) = 0.24  P(dislike|bad) = 2.76e-05  ... |   P(dislike|good) is smaller than P(dislike|bad), but P(y=good) is much bigger than P(y=bad), so the final result is still labeled as “good”. |