

# Programming Assignment 4: Naïve Bayes Text Classification

Assigned: Nov. 18 Due: Dec. 4

Note: Due to the end of the semester, this assignment will only be accepted one week late, to December 11.

In this assignment you will use the Naïve Bayes method to learn a classifier for text from training data, and use a test set to evaluate the quality of the classifier.

## 1 Input format

The major input to the program consists of a file of short biographies (a couple of sentences taken from Knowledge Graph and Wikipedia) of various people, tagged with a category. There are two samples available: a small one at <http://www.cs.nyu.edu/faculty/davise/ai/bioCorpus.txt> and an tiny one at <http://www.cs.nyu.edu/faculty/davise/ai/tinyCorpus.txt>

The format of this file is as follows:

Separate biographies are separated by 1 or more blank lines.

In each biography, the first line is the name of the person. The second line is the category (a single word). The remaining lines are the biography. You may assume that the characters in the file are all a-z, A-Z, comma, period, and white space; there are no numbers and no other punctuation.

The program should have two parameters, supplied on the command line: The first parameter is the file name for the corpus. The second input is a user input of an integer N, which is the number of entries in the corpus to use as the training set. The program will use the first N biographies in the corpus as the training set and the rest as the test set.

Thus a typical command line execution of your program would be

`python3 textClassifier.py corpus.txt 5` or

`java textClassifier corpus.txt 5.`

## 2 Output format

The output contains:

- For each person in the test set, the probabilities associated with each category; the prediction (the category with the highest probability), and a statement whether this is right or wrong.
- The overall accuracy of the classifier on the test set.

For instance, for the tiny corpus linked above, with  $N=5$ , the output is:

Benjamin Disraeli. Prediction: Writer. Wrong.  
Government: 0.44 Music: 0.07 Writer: 0.48

George Eliot. Prediction: Writer. Right.  
Government: 0.01 Music: 0.07 Writer: 0.91

Barbara Jordan. Prediction: Government. Right.  
Government: 0.97. Music: 0.03. Writer: 0.00

Clara Schumann. Prediction: Music. Right.  
Government: 0.01. Music: 0.98. Writer: 0.01

Overall accuracy: 3 out of 4 = 0.75.

Probabilities should be given to an accuracy of 0.01, as above.

This output is available at <http://www.cs.nyu.edu/faculty/davise/ai/tinyOutput.txt>.

I will use this as an example throughout the assignment. The “tiny training corpus” is the first 5 biographies in the tiny corpus, and the “tiny test corpus” is the last 4.

## 3 The Program

The program is divided into two parts: Learning the classifier from the training set (the first  $N$  biographies in the input), and evaluating the classifier over the test set (the remaining biographies).

### 3.1 Learning the classifier from the training set

#### 3.1.1 Normalization

In reading in the training set:

- All words in the biography should be normalized to lower case.
- Stop words should be omitted. Stop words are: (a) any word of one or two letters; (b) any word in the list of stopwords provided. <http://www.cs.nyu.edu/faculty/davise/ai/stopwords.txt>.

You may either read the list of stopwords from a text file “stopwords.txt”, in the same working directory as your program, or you may hard-code the list of stop words in your program.

#### 3.1.2 Counting

Compile a count of

- For each category  $C$ , the number of biographies of category  $C$  in the training set  $T$ ,  $\text{Occ}_T(C)$ .
- For each category  $C$  and word  $W$  in the training set  $T$ , the number of biographies of category  $C$  that contain word  $W$ ,  $\text{Occ}_T(W|C)$ . It does not matter how many times a word occurs within

a given biography, as long as it occurs once (this is known as the “Bernoulli method” or, more generally, as a “set of words” model).

Note: THE CATEGORY NAME DOES NOT COUNT AS A WORD IN THE TEXT. So, for example, in the biography of Bella Abzug, “Government” is not considered a word  $W$  in the biography. However, if “government” occurs in the *text* of a biography then it counts.

### 3.1.3 Probabilities

**Step 1:** For each classification  $C$ , define  $\text{Freq}_T(C) = \text{Occ}_T(C)/|T|$ , the fraction of the biographies that are of category  $C$ . For instance, in the tiny training corpus,  $\text{Freq}_T(\text{Government}) = 2/5$ .

For each classification  $C$  and word  $W$ , define  $\text{Freq}_T(W|C) = \text{Occ}_T(W|C)/\text{Occ}_T(C)$ , the fraction of biographies of category  $C$  that contain  $W$ . For instance, in the tiny training corpus,  $\text{Freq}_T(\text{“american”}|\text{Government}) = 1/2$ , since there are two biographies of category *Government* and one of these has the word “american”.

**Step 2:** For each classification  $C$  and word  $W$ , compute the probabilities using the Laplacian correction. Let  $\epsilon = 0.1$ .

$$P(C) = \frac{\text{Freq}_T(C) + \epsilon}{1 + \text{number of categories} * \epsilon} \qquad P(W|C) = \frac{\text{Freq}_T(W|C) + \epsilon}{1 + 2 * \epsilon}$$

In the second equation, the “2” in the denominator reflects the fact that each random variable  $W$  has two possible outcomes: true and false.

For instance  $P(\text{Government}) = (0.4 + 0.1)/(1 + 3 * 0.1) = 0.3846$ .

$P(\text{“american”}|\text{Government}) = (0.5 + 0.1)/(1 + 2 * 0.1) = 0.5$ .

**Step 3:** Compute negative log probabilities to avoid underflow. The base of the logarithm doesn’t matter, except that you will have to use the same base when you reconstruct the probabilities in step 4.b of part 2. I’ve used base 2 in computing the trace. For each classification  $C$  define  $L(C) = -\log_2(P(C))$  and define  $L(W|C) = -\log_2(P(W|C))$ . For instance,  $L(\text{Government}) = 1.3785$  and  $L(\text{“american”}|\text{Government}) = 1.0$ .

Now you’re done with the learning phase.

## 3.2 Applying the classifier to the test data

For each biography  $B$ :

- Step 1. Normalize the text as in part I. Additionally skip any word that did not appear at all in the training data.
- Step 2. For each category  $C$ , compute  $L(C|B) = L(C) + \sum_{W \in B} L(W|C)$ . As in the learning phase:
  - Each word counts only once, no matter how often it appears in the biography.
  - DO NOT USE THE CATEGORY ATTACHED TO THE BIOGRAPHY. This is even more important now than it was in part I. Whenever you use part of a labelled corpus as a test set, you obviously have to ignore the label; otherwise, the learning algorithm is just cheating off a crib sheet.
- Step 3. The prediction of the algorithm is the category  $C$  with the smallest value of  $L(C|B)$ . It is extremely unlikely that you will ever get a tie; if you do, it may be broken arbitrarily (i.e. don’t worry about the case.)

Step 4. To recover the actual probabilities: Let  $k$  be the number of categories.

- 4.a For  $i = 1 \dots k$  let  $c_i = L(C_i|B)$ , the values of  $L$  for all the different categories. Let  $m = \min_i c_i$ , the smallest value of these.
- 4.b For  $i = 1 \dots k$ , if  $c_i - m < 7$  then  $x_i = 2^{m-c_i}$  else  $x_i = 0$ . The base 2 here is the same one as used in step 3 of the learning algorithm.
- 4.c Let  $s = \sum_i x_i$ , For  $i = 1 \dots k$ ,  $P(C_k|B) = x_i/s$ .

There is a trace of the execution of the algorithm for the tiny corpus with  $N = 5$  at <http://www.cs.nyu.edu/faculty/davise/ai/tinyChart.txt>.

## 4 Program requirements

You may hard code the list of stop words.

You may not hard code the set of categories; e.g., do not assume that the only categories are Government, Music, and Writer. Your code will be tested on a corpus that has other categories.

If you want, you may assume that the entire corpus fits in memory, so it is OK to begin by reading in the whole file `corpus.txt` and then working on it as an in-memory data structure. However, it is preferable, and actually not very difficult, to write the program so that it works by doing a single pass reading through the file; it just requires carefully keeping track of things. The memory required for doing this the right way is  $O(\text{number of different words} * \text{number of different categories})$  which in general is much smaller than the total size of the corpus.

You may assume that the input is correctly formatted, as specified in section 1 of the assignment. However, please write your code so that it is robust relative to trailing blanks at the end of a line, multiple blank lines between biographies, and blank lines at the end of the biography file.