Assignment 4

Information Retrieval and Web Search
Winter 2017
Total points: 80
Issued: 03/09/2017 Due: 03/23/2017

All the code has to be your own (exceptions to this rule are specifically noted below). The code must run on the CAEN environment without additional installation or additional files (except for the data files specified in the assignment).

You can discuss the assignment with others, but the code is to be written individually. You are to abide by the University of Michigan/Engineering honor code; violations will be reported to the Honor Council.

[80 points] Naive Bayes classifier.

Write a Python program that implements the Naive Bayes text classifier, as discussed in class. To avoid zero counts, make sure you also implement the add-one smoothing.

Evaluate your implementation on the deception dataset deception.tar.gz provided on Canvas under the Files/ section.

The dataset consists of 196 files, all of them representing statements about a best friend, half of which are deceptive and half of which are truthful. The ground truth (i.e., label) for each statement is encoded in the filename; for instance, the statement stored in the file lie13 is deceptive.

For evaluation, use the leave-one-out strategy, meaning that you train your Naive Bayes classifier on 195 files, and test on the remaining one file. Repeat this process 196 times.

Programming guidelines:
Write a program called *naivebayes.py* that trains and tests a Naive Bayes classification algorithm. The program will receive one argument on the command line, consisting of the name of a folder containing all the data files.

Include the following functions in *naivebayes.py*:
a. Function that trains a Naive Bayes classifier:
Name: *trainNaiveBayes*; input: the list of filenames to be used for training; output: data structure with class probabilities; output: data structure with word conditional probabilities; output: any other parameters required (e.g., vocabulary size).

Given a set of training files, this function will:
- preprocess the content of the files provided as input, i.e., apply tokenizeText. You are encouraged to use the functions you implemented for Assignment 1. In the basic implementation, do not remove stopwords, do not use stemming. See the Write-up guidelines for required variations.
- calculate all the counts required by the Naive Bayes classifier.

b. Function that predicts the class (truth or lie) of a previously unseen document.
Name: *testNaiveBayes*; input: the filename to be used for test; output: predicted class (truth or lie)

The main program should perform the following sequence of steps:
i. open the folder containing the data files, included in the folder provided as an argument on the command line (e.g., *bestfriend.deception/),* and read the list of files from this folder.
Repeat 196 times:
    i. select one file as test, and the remaining as training.
    ii. apply the trainNaiveBayes followed by the testNaiveBayes functions.
    iii. determine if the class assigned by the testNaiveBayes function is correct.

The *naivebayes.py* program should be run using a command like this:
% *python naivebayes.py bestfriend.deception/*

It should produce a file called *naivebayes.output* consisting of pairs of file names, along with the class labels predicted by your implementation. E.g.:
lie1.txt lie
lie2.txt true
true1.txt true
true2.txt true
etc.

It should also display (standard output) the accuracy of your classifier, calculated as the total number of files for which the class predicted by your implementation coincides with the correct class, divided by the total number of files.

[If necessary, you can add extra arguments and/or extra return values to the functions.
Please don't use scikit-learn or any other python library that implements the Naive Bayes classifier.]

Write-up guidelines:

Create a file called *naivebayes.answers*. Include in this file the following information:
1. accuracy of your Naive Bayes classifier, as described above
2. accuracy of your Naive Bayes classifier, when you also remove the stopwords

3. accuracy of your Naive Bayes classifier, when you also stem the words
4. accuracy of your Naive Bayes classifier, when you also remove the stopwords and stem the words

5. Using the implementation that does not remove stopwords and does not stem words, list the top 10 words that have the highest conditional probability (i.e., P(w|c)) in each of the two classes considered (truth, lie). Under each class, list the words in reversed order of their conditional probability (i.e., descending order).

General Canvas submission instructions:

- Include all the files for this assignment in a folder called *[your-uniqname].Assignment4/* **Do not** include the data folder, i.e., *bestfriend.deception.*
  For instance, mihalcea.Assignment4/ will contain naivebayes.py (please include the initial implementation that does not remove stopwords and does not stem words), naivebayes.output, naivebayes.answers.
- Archive the folder using tgz or zip and submit on Canvas by the due date.
- Make sure you include your name and uniqname in each program and in the answers file.
- Make sure all your programs run correctly on the CAEN machines.