Assignment 4

Information Retrieval and Web Search
Winter 2022
Total points: 90
Issued: 03/11/2022
Due date 1: 03/16/2022
Due date 2: 03/24/2022

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.

[25 points] Due 03/16/2022

Misinformation data collection

Any text classification project starts with data collection. For the first part of this assignment, you will have to identify online true information and online misinformation in English on one topic: the Ukraine-Russia war or Climate change. (We understand that some students may be directly affected by the war; if that is the case, please select the Climate change topic for the data collection and later for the experiments you will run).

Please use this form to submit true information and misinformation you identify online.
https://docs.google.com/forms/d/e/1FAIpQLSfKTaaY6BRqHq5v75J3xQ1BuVdWIdDjcuqNEn1BIvDH7Ronhg/viewform

These are the steps you will have to follow to complete the form:

- First, please select the topic for which you will be submitting data: the Ukraine-Russia war, or Climate change.
- Next, on the topic selected, please identify 7 pieces of true information and 7 pieces of misinformation, all written in English. Any English online text source is acceptable, eg, news sources, Twitter, blogs, etc. Regardless of the source, please make sure each piece of (mis)information is at least 200 characters long and at most 1000 characters long. Please also keep track of and record the URL from where you are getting the (mis)information.
- For at least 3 pieces of information and at least 3 pieces of misinformation, please also write an explanation indicating why you believe the information is truthful (or misinformative); for the remaining cases where you do not provide an explanation, please include NA.

- Consider saving the data prior to completing this form, to make sure you don't lose it if something goes wrong with the form submission.

[65 points] Due 03/24/2022

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:
(1) the fake news dataset `fakenews.zip` provided on Canvas under the Files section;
(2) one of the ClimateChange or UkraineWar datasets (your choice), which will be made available on 3/17

The first dataset consists of the folder `fakenews/` that contains 480 files, consisting of fake and legitimate news, covering several domains (technology, education, business, sports, politics, entertainment and celebrity news). The ground truth (in this case also referred to as the label or the class) for each statement is encoded in the filename; for instance, the statement stored in the file `fake13.txt` is misinformative (the class is *fake*).

For both evaluations, use the *leave-one-out* strategy. That is, assuming there are N files in a dataset, train your Naive Bayes classifier N-1 files, and test on the remaining one file. Repeat this process N times, using one file at a time as your test file.

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 file paths to be used for training;
    output: data structure with class probabilities (or log of probabilities);
    output: data structure with word conditional probabilities (or log of 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., tokenize the text (you are encouraged to use the functions you implemented for Assignment 1, but you can also use another CAEN-compatible tokenizer if you prefer; please also include the related code if you choose to use your own processing code) and compute a vocabulary (which is composed of all the tokens in the training files, regardless of what class they belong to). In the basic implementation of your function,

do not remove stopwords, do not use stemming. See the Write-up guidelines below for required variations.
- calculate all the counts required by the Naive Bayes classifier
- your implementation should be the multinomial Naive Bayes

b. Function that predicts the class (true or fake) of a previously unseen document:

name: `testNaiveBayes`;
input: the file path to be used for test;
input: the output produced by *trainNaiveBayes*;
output: predicted class (the string "*true*" or the string "*fake*". You can assume these to be the only classes to be predicted)

The tokens that are not in the vocabulary should have smoothing applied.

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., `fakenews/`), and read the list of files from this folder. Assuming the folder has N files (e.g., fakenews/ includes 480 files).

Repeat N times:

i. select one file as test, and the remaining N-1 as training (ex: the first iteration should use the first file as testing; the second iteration should use the second file as testing; and so on.)
iteration).
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* [*datafolder/*]

(e.g., $ *python naivebayes.py* fakenews/)

It should produce a file called `naivebayes.output.[datafolder]` (e.g., `naivebayes.ouput.fakenews` for the `fakenews/` datafolder). The file should consist of pairs of file names along with the class labels predicted by your implementation. The order of files in this output does not matter. E.g.:

```
true1.txt    fake
true2.txt    true
fake1.txt    true
fake2.txt    fake
```
...

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 do not 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 on the fakenews dataset, as described above
2. Accuracy of your Naive Bayes classifier on the second dataset (ClimateChange or UkraineWar)
3. For each of the two datasets, find and include the top 10 words that have the highest conditional probability (i.e., P(w|c)) in each of the two classes considered (true, fake). Include both the words as well as the conditional probability. This can be from any of the N test iterations that were executed. Under each class, list the words in descending order of their conditional probability.

General Canvas submission instructions:

- Include all the files for this assignment in a folder called `[your-uniqname].Assignment4/` **Do not** include the data folders..
  For instance, `mihalcea.Assignment4/` will contain `naivebayes.py`, `naivebayes.output.fakenews`, naivebayes.output.[seconddataset], `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 without the need to install libraries that are not already available on CAEN.