DAT405 Introduction to Data Science and AI

2019 – 2020, Reading period 1

Assignment 4: Spam classification using Naïve Bayes

There will be an overall grade for this assignment. To get a pass grade (grade 3), you need to pass items 1-3 below. To receive higher grades, finish items 4 and 5 as well.

In this assignment you will implement a Naïve Bayes classifier in Python that will classify emails into spam and non-spam ("ham") classes. Your program should be able to train on a given set of spam and "ham" datasets, including counting the frequency of each *token* (i.e. word) in both data sets.

You will work with the datasets available at https://spamassassin.apache.org/old/publiccorpus/. There are three types of files in this location:

- easy-ham: non-spam messages typically quite easy to differentiate from spam messages.
- hard-ham: non-spam messages more difficult to differentiate
- spam: spam messages

Read the "readme.html" for a full description of the file contents. The .bz2-file can be unzipped using the linux command

```
tar –xjvf file.bz2
```

which will result in a directory named "file", containing all email messages as separate files.

1. Preprocessing:
   a. Note that the email files contain a lot of extra information, besides the actual message. Ignore that for now, and run on the entire text. Further down (in the higher grade part), you will be asked to filter out the headers and footers.
   b. We don't want to train and test on the same data. Split the spam and the ham datasets in a training set and a test set in separate folders.
2. Write a Python program that:
   a. Takes the four datasets (hamtrain, spamtrain, hamtest, and spamtest) as input.
   b. Trains on the training sets using Maximum Likelihood. Uses Laplace add-one smoothing to avoid zero-counts of any tokens. [*Tip: to avoid working with too low probabilities, use log probs and sums instead of probabilities and products.*]
   c. Runs a Naïve Bayes classifier on the test sets and outputs the classification results to the screen, including
      - The number of emails in each of the four data sets.
      - The percentage of the ham and spam test sets that were classified correctly.
3. Run your program on
   i. Spam versus easy-ham
   ii. Spam versus hard-ham

   and include the results in your report (see below).

4. To avoid classification based on common and uninformative it is common to filter these out.
    a. Argue why this may be useful. Discuss how this could be done. Here's a list of common words, also known as *stop words*, that might be useful https://algs4.cs.princeton.edu/35applications/stopwords.txt.
    b. Choose a method to filter out common words and include it in your code. Run the updated program on your data and record how the results differ from 3.
5. Filter out the headers and the footers of the emails before you run on them. The format may vary somewhat between emails, which can make this a bit tricky, so perfect filtering is not required. Run your program again and answer the following questions:
    a. Does the result improve from 3 and 4?
    b. What are the five most frequent tokens in spam and ham messages respectively. Was there any difference in these lists before and after using the stop words filter?
    c. The split of the data set into a training set and a test set can lead to very skewed results. Why is this, and do you have suggestions on remedies?
    d. What do you expect would happen if your training set were mostly spam messages while your test set were mostly ham messages?

What to submit

- All Python code written.
- A report stating
    - Your names and results and how many hours each person spent on the assignment.
    - Model description and the results on the runs.
    - Answers to questions in 4 and 5.

Make sure to give the names of all the people in the group on all files you submit!

If you upload a zip file, please also upload any PDF files separately (so that they can be viewed more conveniently in Canvas).

Deadline: Monday 30 September 2019 at 12:00 (noon).