DAT405 Introduction to Data Science and AI, SP2 2021

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.

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

Download the three dataset files that have names starting with 20021010.

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. On Windows you can use 7-Zip (https://www.7-zip.org/download.html) to decompress the 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 into a training set and a test set. (hamtrain, spamtrain, hamtest, and spamtest).
2. Write a Python program that:
   a. Uses four datasets (hamtrain, spamtrain, hamtest, and spamtest)
   b. Trains a Naïve Bayes classifier (e.g. from Sklearn) on the training sets (hamtrain and spamtrain), then classifies the test sets (hamtest and spamtest) and reports the percentage of ham and spam test sets that were classified correctly. You can use CountVectorizer to transform the email texts into vectors. Please note that there are different types of Naïve Bayes Classifier in SKlearn (documentation is available here). Test two of these classifiers: 1. Multinomial Naive Bayes and 2. Bernoulli Naive Bayes that are well suited for this problem. For the case of Bernoulli Naive Bayes you should use the parameter ***binarize*** to make the features binary. Discuss the differences between these two classifiers.
3. Run your program on
   i. Spam versus easy-ham
   ii. Spam versus hard-ham

   and include the results in your report.

4. To avoid classification based on common and uninformative words it is common to filter these out.
    a. Argue why this may be useful. Try finding the words that are too common/uncommon in the dataset.
    b. Use the parameters in Sklearn's CountVectorizer to filter out these words. Run the updated program on your data and record how the results differ from 3. You have two options to do this in Sklearn: either using the words found in part (a) or letting Sklearn do it for you.
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. 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?
    c. 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.
    - The results on the runs.
    - Answers to questions.

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: Tuesday 30 November 2021 at 23:59.

Self-check

Is all the required information on the front page? Have you answered all questions to the best of your ability? Anything else you can easily check? (details, terminology, arguments, clearly stated answers etc.?)

Do not submit an incomplete assignment! We are available to help you, and you can receive a short extension if you contact us.

Grading

Grading will be based on a qualitative assessment of each assignment. It is important to:

- Present clear arguments
- Present the results in a pedagogical way
    - Should it be table/plot? What kind of plot? Is everything clear and easy to understand?

- Show understanding of the topics
- Give correct solutions.
- Make sure that the code is well commented.
    - Important parts of the code should be included in the running text and the full code uploaded to Canvas.