CSC484 | Project 2

In this project we will build a multinomial naive Bayes email spam classifier that would classify emails to spam or ham (not spam).

Data

We will use an email dataset that contains spam and ham emails. The dataset can be downloaded from LMS. Here's a description of the data:

Directory name	Description	Use
SPAM_training_set	Contains >18,000 email. Spam emails start with "SPAM." and ham emails start with "HAM."	Use it for training your model
SPAM_test_set	Contains 800 email. Spam emails start with "SPAM." and ham emails start with "HAM."	Don't use it for training. Use it to test your model and evaluate it.
SPAM_toy	Contains 8 toy examples that follow the same format as the previous directories. These are not real emails.	Use them when developing your model. It is easier to find bugs in your code when you're dealing with a small dataset. Using this dataset is optional. You're not required to use it; it's just there in case you need it.

Part (a): Training

You are asked to create a function that allow us to train a naive Bayes classifier on the spam dataset provided. Please implement in python the function:

`nb_train(x, y)`

The function will accept two parameters: **x** (a list of documents/emails) and **y** (a list of integers where 0 means HAM and 1 means SPAM). You can assume that the list **y** will not contain any values other than 1 or 0. You may also assume that **x** and **y** will always have the same length.

The function will return a trained model (dictionary) with the following format:

```
model = {

 'ham_count':  # number of HAM emails

 'spam_count': # number of SPAM emails

 'ham_fd':     # a dict containing the frequency distribution
               # of words/features in the HAM emails.
               # ham_df['the'] should return the number of times
               # the word 'the' occurs in the HAM emails.
```

```
 'spam_fd':    # a dict containing the frequency distribution
               # of words/features in the SPAM emails.
               # spam_fd['the'] should return the number of
               # times the word 'the' occurs in the HAM emails.
}
```

Part (b): Testing

You are asked to create a function that allow us to test a trained naive Bayes classifier on new emails/documents we haven't seen.

Please implement in python the following function:

`nb_test(docs, model, use_log=False, smoothing=False)`

where **`model`** is the trained model in part (a) and **`docs`** is a list of documents/emails that we need to classify. The **`use_log`** argument specifies whether to calculate by multiplying the probabilities or summing the log of the probabilities, and **`smoothing`** will determine whether to smooth the probabilities by applying add-one smoothing or not. The function will return list of 0s and 1s. The *i*-th position in the list will be 0 if the *i*-th email in the list **`docs`** is classified as HAM and 1 if the email is classified as SPAM. The returned list must be the same length as the **`docs`** list.

Part (c): Experiments and evaluation

Create a trained model with the provided training data using the function you implemented in part (a). Test the emails in the test set using four configurations (with/without log calculation, and with/without smoothing) and report the f1-score of each configuration. In order to do so, you'll need to create a function to calculate the f-score

`f_score(y_true, y_pred)`

where **`y_true`** is a list of the true/real/golden labels provided in the test set and **`y_pred`** is the list of the classifications returned by your classifier. Both lists will only contain 0s and 1s and should have the same length. The function will return a single value: the f-score of your classifications.

Deliverables

There are two deliverables for this assignment:

1) You should submit a .py file containing all the code you implemented
2) You should submit a pdf deck/presentation showing:
   A. The results of the part (c) experiments
   B. Some of the design choices you made and why

The deck should be in a presentation format and should not be done using a word processing software. The deck should be carefully designed and prepared using a presentation software (Microsoft PowerPoint, Apple Keynote, etc.).

Notes:

- You are not allowed to use any external libraries. Basic text processing (tokenization, stemming) using NLTK is fine. However, other NLTK capabilities are not allowed. If you are in doubt, please ask me before using any external library.
- You are allowed to create as many helper functions as needed.
- You may find the attached `project2skeleton.py` file useful. You don't need to use it but it's there if you want it.