SPAM or HAM

In this assignment, you will build a spam classifier from scratch. No training data will be provided. You are free to use whatever training data that is publicly available/does not have any copyright restrictions (You can build your own training data as well if you think that is useful). You are free to extract features as you think will be appropriate for this problem. The final code you submit should have a function/procedure which when invoked will be able to automatically read a set a emails from a folder titled test in the current directory. Each file in this folder will be a test email and will be named 'email.txt' ('email1.txt', 'email2.txt', etc). For each of these emails, the classifier should predict +1 (spam) or 0 (non Spam). Two sample emails your classifier will be tested on can be found in the folder test. You are free to use whichever algorithm learnt in the course to build a classifier (or even use more than one). The algorithms (except SVM) need to be coded from scratch. Your report should clearly detail information relating to the data-set chosen, the features extracted and the exact algorithm/procedure used for training including hyperparameter tuning/kernel selection if any. The performance of the algorithm will be based on the accuracy on the test set.

Solution:

Kindly run the file **testSPAM.m** to get the outut file which contains 1 for spam email and zero for ham email.

Caution: As my dictionary is quite large so testSPAM.m may take 5 to 10 minutes for giving the output depending on the number of test data and words in each test data and system computational power so be patient for my test set containing 260 mails it took around 3.5 minutes

Its a binary classification problem. There can be various ways to solve it.I am going with **Naive Bayes** modelling.

Background Intution:

By Bayes Theorem we have:

$$P(SPAM|X) = \frac{P(X|SPAM)*P(SPAM)}{P(X|SPAM)*P(SPAM)+P(X|HAM)*P(HAM)} ------------------(i)$$

$$P(HAM|X) = \frac{P(X|HAM)*P(HAM)}{P(X|SPAM)*P(SPAM)+P(X|HAM)*P(HAM)} ---------------(ii)$$

where X is the vector of words contained in the email. we are going to predict it is spam or not ,also let us say X has N words

Let us use Multinomial Bayes (assume occurrence of one word does not affect the occurence of other.

Then $P(X|Anything) = \prod_{i=1}^{N} P(X_i|Anything)$

Clearly denominator of the RHS of equation (i) and (ii) are same and hence can be treated as constant .

So

$P(SPAM|X) \propto P(X|SPAM) * P(SPAM)$

and

$P(HAM|X) \propto P(X|HAM) * P(HAM)$

Now the question is we will be given a set of mails with labels as SPAM or HAM then how to use it to calculate the probabilities ?
To solve this what we will do is for each email irrespective of SPAM or HAM preprocess the contents of the emails.
In preprocessing step we will do the following:
• Lower-casing: The entire email is converted into lower case, so that captialization is ignored (e.g., IndIcaTE is treated the same as Indicate).
• Stripping HTML: All HTML tags are removed from the emails. Many emails often come with HTML formatting; we remove all the HTML tags, so that only the content remains.
• Normalizing URLs: All URLs are replaced with the text "httpaddr".
• Normalizing Email Addresses: All email addresses are replaced with the text "emailaddr".
• Normalizing Numbers: All numbers are replaced with the text "number".
• Normalizing Dollars: All dollar signs ($) are replaced with the text "dollar".
• Word Stemming: Words are reduced to their stemmed form. For example, "discount", "discounts", "discounted" and "discounting" are all replaced with "discount". Sometimes, the Stemmer actually strips off additional characters from the end, so "include", "includes", "included", and "including" are all replaced with "includ".
• Removal of non-words: Non-words and punctuation have been removed. All white spaces (tabs, newlines, spaces) have all been trimmed to a single space character.

In preprocessing step only i am collecting all the words occuring in each email .
Then i am creating a dictionary of most frequent words

Next task is to create feature matrix M
where M(i,j,k) represents ith traing example containg the word of rank j in the dictionary and having a frequency k

Ones our feature matrix is ready ,we can proceed for calculating the probabilities
First we store the indices of the indices of spam and non spam messages in a vector $train\_labels$.
Then P(spam)=Number of spam email/Total number of email
i.e. prob_spam = length($spam\_indices$) / numTrainDocs;
similarly
$prob\_ham$ = length($ham\_indices$) / numTrainDocs;
Then we find the number of words in each email and store it in email_length

Then we find word_count for spam and ham mails
ones we have that we can easily calculate the probability of a mail given that we know the toens in it using the bayes formula as written above i. formula 1 and 2

Explanations of functions used for training the model.
I have trained the model very extensively and took data from various sources and have

combined them.
First data set i took can be found here
https://archive.ics.uci.edu/ml/machine-learning-databases/00228/ It contains 5574 mails which are tagged according being ham (legitimate) or spam.
Other two data set that i used are a mixture from here and there and can be find in the folder named spam-train and nonspam-train each of which contains 480 mails. So total number of training docs=numTrainDocs=5574 + 480 + 480 = 6534

quite huge data set for better accuracy!!
First dataset i saved as SpamCollection5574.xlsx and loaded it into matlab. Then i made a function **WordArray.m** which processes the contents of each email and collect all their processed words in the array named **words**, which will be used further to create dictionary.
First version of the dictionary is created using the function **createDictionary.m** which stores top 2500 frequent words from the word array.
Also i collected a file called vocab.txt and merged it with dictionary1 and kept the unique words only to the final version of dictionary

Then i created the feature matrix M using the function **createFeature.m** Feature matrix M contains the frequecy of each word that are present in the dictionary corresponding to each email.

Similar thing i repeated for the other two data set. For this i took the help of the function **processEmailMy.m** which performs similar task as **createFeature.m**
ones the feture matrix was ready calculating probabilities becomes trivial and simple formulae of probabilities i calculated the respective probabilities.
See comments in the code **trainSpam.m**for clear idea.

Once our model is trained. On my ciomputer with 4GB rRAM intel 5 2nd generation it took around 20 minutes for training then i saved the workspace in a file named **trainedWorkSpace.mat**

Now time for testing
For testing i used the function **testSPAM.m**
I tested it on 260 mails contained in two folders namely folder nonspam-test spam$_testandtheirlabesisstore$
$labels.txt, igotanaccuracyof 98.077\% Kindlychechkthecodechek.m forthisIampurposefullyommitingthe$

Explanation for the **testSPAM.m**
I first of all loaded the trained workspace contained in the file 'trainedWorkSpace.mat'
Then change the directory address in such a way that it will point to the test folder in the current directory
As done for training data ,similarly for test data also i will be creating feature matrix with the help of **processEmailMy.m**

And then we know all the required parameters just we need to calculate Probability of a given mail being spam provided we know the tokens(processed words in the mail) it contains.
since we are using multinomial naive bayes so probaility of mail provided tokens = product of probabilities of mail provided individual tokens and since individual probabilities can be very small so their product may cause underflow or over flow that why i used log to compare which probability is higher if if represnting probability of a test mail being spam in the variable log_a and that being ham in the variable log_b when log_a is greater i am outputing 1 onto the screen else 0 onto the screen and at the end writing a text file named **output.txt** that contains zeros and ones corresponding to each email beaing ham or spam respectively

Kindly run the file **testSPAM.m** to get the outut file which contains 1 for spam email and zero for ham email.
Caution: As my dictionary is quite large so testSPAM.m may take 5 to 10 minutes for giving the output depending on the number of test data and words in each test data and system computational power so be patient for my test set containing 260 mails it took around 3.5 minutes