Horia “Ryder” Stancescu
hstances - 0721385

Assignment 5

Spam vs Ham

Step 1:

I collected ~30 messages from my spam filter, and ~30 from my inbox. 10-20 of the messages from each were placed in a “Ham” and “Spam” folder respectively, while the remaining amount were placed in a “NotUsed_Ham” and “NotUsed_Spam” folder. These extra two folders are for step 11 when evaluating the accuracy of the algorithm comes into play.

Step 2:

This task was accomplished by first deciding on what are relevant delimiter tokens. Thus a token for this task is relevant if it does not belong in a word, thus punctuation, mathematical symbols, newlines, tabs, numbers, and spaces are all valid delimiters. Essentially anything that is not a character. This is accomplished by using pythons built in regex split command and supplying this line of regex which pertains to my definition of a relevant token [^a-zA-Z].

Step 3:

This task is accomplished by using pythons efficient use of dictionary structures and adding the words as keys while assigning a value of 1 each time a key is added. If a key already exists in the dictionary structure when a word is read, then simply grab its value, increment by one and reassign the key.

Step 4:

This step was more an instruction for step 5 than anything else, therefore any explanation on this step will be explained in step 5.

Step 5:

Estimating the amount of email that is spam vs not is quite simple. The reason this is simple, is my accounts do have spam filters installed, however I do not (out of negligence) clear or delete my spam filter. When I do, I usually clear my inbox as well resulting in a clean slate. With this in mine my inbox currently has about 1,000 messages, while my spam box has about 400 (rounded to the nearest hundred for mathematical simplicity). Therefore to obtain my probabilities of a message being Spam it’s simple

my Spam Messages /Message Total, and likewise for non-spam IE “Ham”, Ham / Message Total.
Spam = 400/ 1400 = 29%
Ham = 1000/ 1400 = 71%
Using these probabilities the likelihood of a new message being spam vs ham is 29% vs 71%.

Step 6:

Bayes Theorem spam filtering is P(Spam | word), in other words it calculates the probability an email is spam given a keyword is present within its contents. So how is this calculated exactly? To calculate this we first need to acquire (ahead of time) a set of keywords along with their probabilities of occurring in a spam and ham email (step 2 & 3). Now because we have the frequency of a certain word occurring inside a dataset we marked as spam, by dividing that words frequency by the total frequency we arrive at a probability of the word occurring inside a spam email in other words P(Spam | word).

Step 7:

Since an email can be either spam or ham not both, we’ll treat these percentages as independent from one another. Thereby to calculate P(spam | word1 and word2 and word3…), we simply add the probability of each word occurring in a spam message, obviously the more keywords that exist inside a spam message the higher the likelihood of it being spam, the same application is done for ham.

Step 8:

For the Bayesian filter, we simply create a dictionary of the words inside any specific email, then using step 7 add the probabilities of each keyword that’s hit. If the probability of it being spam is higher than ham, then we mark it as spam, and thus the filter is complete.

Step 9:

If a message has a higher likelihood of being spam vs ham then it should be marked as spam. If eliminating false positives is a concern then simply append more data onto the filter’s spam and ham datasets. This is how a Bayesian filter operates and improves, with each email the user marks as spam, the filter improves and thus gets individualized per user, giving spammers a more difficult time in sending the message. It is because of this core functionality I do not feel a need to modify the Bayesian filter as a larger dataset obviously improves the performance in detection, and as this is a test environment written in python, I feel the filter performs adequately.

Step 10:

How well did the algorithm perform on its own filtered datasets.

Email Name	Email:Spam/Ham	Algorithms Decision
Another insane dare	Spam	Spam
cheatingwomen	Spam	Spam
foxybbw	Spam	Spam
fw hi there	Spam	Spam
hello there	Spam	Spam
hey there(2)	Spam	Spam
hey there(3)	Spam	Spam
hey there	Spam	Spam
hey! Remember me	Spam	Spam
hey	Spam	Spam
I am real & its 100% sure	Spam	Spam
lets work it out	Spam	Spam
lorrainedickerson	Spam	Spam
megafbook	Spam	Spam
melissa barnett	Spam	Spam
misti platz	Spam	Spam
no strings fun	Spam	Spam
no subhect	Spam	Spam
online personals	Spam	Spam
sara angelica	Spam	Spam
secretfling22	Spam	Spam
weekendkisses	Spam	Spam
10newjobs	Ham	Ham
checkpoint2	Ham	Ham
courselink	Ham	Spam
cpesstudentcouncil	Ham	Ham
dcalvert	Ham	Ham
dg_nom2	Ham	Ham
dg_nom3	Ham	Ham
dg_nominated	Ham	Ham
dgillis	Ham	Spam
dmccaugn	Ham	Ham
lootcrate	Ham	Ham
minionmeeting	Ham	Ham
vevo2	Ham	Ham

In regards to recognizing it’s own spam it got 100% correct. However in regards to recognizing its spam it got 11/13 which is ~85%. You’ll notice however of the two marked as spam dgillis and courselink the reasons are primarily due to message length. As spam now a days or at least in my inbox seem to appear as though they were written by people (smarter spam), the constant factor is message length. As such fewer keyword matches indicate a potentially higher possibility of the message being spam if the spam side of the filter P(spam | word1…wordN) has more hits than the ham side P(ham | word1…wordN). In general I would say I’m pleased with the algorithms ability to filter on its own filters, now let’s see how well it does on non-filtered datasets.

Step 11:

Email Name	Email:Spam/Ham	Algorithms Decision
1trickforgettinglaid	Spam	Ham
3wordsthatmakeherhorny	Spam	Ham
adpartner	Spam	Spam
bbwdixiefun22	Spam	Spam
bootycall	Spam	Spam
howarethings	Spam	Spam
marciewaller	Spam	Spam
masoncohen	Spam	Ham
speeddating	Spam	Spam
thistrickmaekseyoungwomenf--kyou	Spam	Spam
thisvideogetsyoulaid	Spam	Ham
forcestratgaming	Ham	Ham
gregklot	Ham	Ham
midtermreport	Ham	Ham
mwirth	Ham	Ham
provost	Ham	Ham
sk	Ham	Ham
steamstore	Ham	Ham
stefank	Ham	Ham
studentaffairs	Ham	Ham
tournament 11 grade	Ham	Ham
tribalhollywood	Ham	Ham
tutoring	Ham	Spam
vevo	Ham	Ham

As you can see with the unfiltered results, the spam filter is quite efficient in detecting new Ham email as Ham, however it seems to also suffer a bit in allowing a bit of spam through.

In regards to detecting spam it performed: 8/11 = 72%
In regards to detecting ham It performed: 12/13 = 92%.

In general I would say these results are quite good, considering the spam I used is slightly different to the spam I used in my filters perhaps leading to the Ham triggers. However the algorithm performs quite well in detecting ham in most cases, with a success rate of 92% in this regard. My main fear was the algorithm being too strict and detecting numerous Ham as spam given the “controlled” conditions using the filter, however it appears to be working quite well at least with the datasets I have used. As I mentioned in step 9, in order to improve performance, the algorithm simply requires larger and larger datasets. As every user has specific tastes, and varying website visits, user targeted ads will be more ineffective on a user who has accumulated a filter against these types of ads. This is how a Bayesian spam filter works in real life, and is what I attempted to mimic in this quick Assignment, simply by adding data to the datasets, the algorithms frequency analysis improves and as such so does its hit and miss rate on Ham vs Spam. This concludes my report on A5.