### **Overview**

##### **In this project, you will employ Bayes Theorem and Bernoulli principle for automatically classifying if an email message is a spam or not.**

Given a dictionary of words [w1, w2, ..., wn] , an email message can be represented using a "feature vector" comprising of several binary elements [f1, f2, ..., fn] . A binary element fi takes value 1 if the corresponding word wi in dictionary is present and is 0 if the word wi is not present. For example, let a dictionary constitute of the following words: "internet", "free", "business", "email", "people". Then an email can be represented using a binary feature vector of [1 0 0 1 1] which indicates this email contains the word "internet", "email" and "people". Each email will also have an associated binary label indicating if email is spam or not. For example, 1 will indicate email is *spam* and 0 will indicate email is *ham* (not spam).

Now, let **D** represent an email represented in form of a bit string (binary feature vector) = [f1, f2, ..., fn] where each fi is either 0 or 1 indicating the presence or absence of word wi.  
Let **C** represent the "class" of the email. That is whether it is spam or not. There are two classes: **C = S** indicates ***spam*** and **C = H** indicates that email is ***ham*** (not spam).

The task of Spam Classification then reduces to figuring out which probability out of **p(S | D)** and **p(H |D)** is higher. If p(S | D) > p(H |D), then the email can be classified as *spam*, otherwise it can be classified as *ham*.

##### **How to compute these "posterior" probabilities**

Consider the probability p(C | D). Using Bayes Theorem, we get

p(C|D) ∝ p(D|C)p(C).

The proportional sign is there because the denominator is common between p(S | D) and p(H | D) and can be factored out when comparing the two.

**Estimating p(C):**p(C) is what is called the *prior* probability (that is, how likely an email message is spam/ham without observing any evidence). This is typically determined by empirical evidence such as the number of email messages that are spam/ham divided by total number of email messages.

**Estimating p(D | C)**: To estimate the *likelihood*, **p(D | C)**, we will use the "Naive Bayes" assumption (this is a new term but it is the same as assumption of  conditional independence between events that we have spoken about in class such as independence between sequence of coin tosses). In particular for this problem, the Naive Bayes assumption states that the probability of each word occurring in the email is independent of the occurrences of the other words (This assumption is often incorrect but has shown remarkable results in classification tasks). This means we can write the document likelihood P(D | C) in terms of the individual word likelihoods P(wt |C) where P(wt | C) is the probability of word wt occurring in a email of class C; the probability of wt not occurring in a email of this class is given by (1 − P(wt | C) ). If we think of each wt as a coin toss with outcome as heads or tails (heads if word is occurring in email of class C, tails if not), then using the Bernoulli principle **P(D | C) = ∏ ftP(wt |C) + (1- ft)(1-P(wt |C)**, where the product (t = 1, 2, ..., n) is over all the words in the dictionary. If word wt is present, then ft = 1 and the required probability is P(wt|C); if word wt is not present, then ft=0 and the required probability is 1−P(wt|C).

**Estimating** **P(wt |C):** This can be estimated from a "training" set of emails of a specific class C (Recall C is either S or H; so we will need to estimate both **P(wt | C = S)**and**P(wt | C=H)**) by measuring the relative frequency of emails of class C that contain word wt. That is, the number of emails of the class C in which the word wt occurs divided by total number of emails of the class C.

##### **Summary**

Given a training set of emails (each labelled with a class), and a set of K classes, we can estimate a Bernoulli spam classification model as follows:

1. Define the dictionary (will be provided in this project); the number of words in the dictionary defines the dimension of the feature vectors
2. Count the following in the training set:

• N the total number of emails

• Nk the number of emails labelled with class C=k, for k=S, H (That is, number of emails known to be spam and ham, respectively)

• nk (wt ) the number of emails of class C = k containing word wt for every class and for each word in the dictionary (That is, frequency of occurrence of word wt in spam/ham emails)

1. Estimate the likelihoods p(wt | C = k)  = nk (wt ) / Nk
2. Estimate the priors p(C = k) = Nk/ N

To classify an unlabelled email Dnew, we estimate the posterior probability for each class **p(C = k | Dnew)** for k = S, H using the equation

p(C|Dnew) ∝ p(Dnew|C)p(C)

= p(C) **∏ ftp(wt |C) + (1- ft)(1-p(wt |C)**

and determine which out of **p(C = S | Dnew)** and **p(C = H | Dnew)**is higher.

**Note**: Most of the material described above is taken from [this resource](https://canvas.sonoma.edu/courses/30798/files/2478014/preview). You are encouraged to consult and review it. Here is another resource that explains [text classificationLinks to an external site.](http://sebastianraschka.com/Articles/2014_naive_bayes_1.html) very well.

#### **Checkpoint A, 75 points (due as an in-person demo by Mon, May 02):**

Implement Bernoulli Naive Bayes classification model as described above and test it on toy example provided in [this resource](https://canvas.sonoma.edu/courses/30798/files/2478014/preview). Your program should read training data and corresponding class information from these 2 files: [toy.data.txt](https://canvas.sonoma.edu/courses/30798/files/2478016/preview) and [toy.class.txt](https://canvas.sonoma.edu/courses/30798/files/2477981/preview). Note that the file toy.class.txt file contains labels 1 and 0, where 1 indicates the class **S** and 0 indicates the class **I.** Use the training data to compute prior probabilities p(S) and p(I) and to compute the likelihood probabilities p(wt | C = k) where k = S, I and t = 1 through 8 (because the toy example has 8 words).

Your program should then read two test cases from [toytest.data.txt](https://canvas.sonoma.edu/courses/30798/files/2478042/preview) and output the posterior probability p(S | Dtest) and p(I | Dtest) for each of them, as follows:

**Test case 1  
 p(S|b) = 0.00561167  
 p(I|b) = 9.30909e-06  
Test case 2  
 p(S|b) = 0.000280584  
 p(I|b) = 0.00804306**