CS 4265 Homework #5

In this assignment, you will write a Scala program that implements the Naïve Bayes classification for email spam identification using the dataset spam_ham_dataset.csv, download from the software repository in D2L. First, we will process a labeled dataset that contains email text. The text must be cleaned up. Remove "subject" and punctuations.

Then compute prior and likelihood based on Bayes theorem.

$p(s|M) = p(M|s)p(s)/p(M)$ where the posterior $p(s|M)$ the probability of an email, represented by M that is a spam. $p(s)$ is the probability of an email that is a spam. We don't need to compute $p(M)$ as we are comparing the relative magnitude of $p(s|M)$ to $p(h|M)$, which is the probability of a given email M that is not a spam or a ham.

$p(h|M) = p(M|h)p(h)/p(M)$

Quantities you need to compute are listed as follows:

1) $p(s)$: # of spam emails in the training dataset divided by the total # of emails in the training dataset.
2) $p(h)$: # of non-spam emails in the training dataset divided by the total # of emails in the training dataset. $p(s) = 1 - p(h)$
3) $P(M|s) = p\left(w_1^{r_1}, w_2^{r_2}, \dots, w_n^{r_n}|s\right) = p(w_1|s)p(w_2|s)\cdots p(w_n|s)r_1 r_2 \cdots r_n$ where $w_i^{r_i}$ indicates that w_i appears in the email M r_i times and $p(w_i|s)$ is the probability of the word w_i in spam email of the training dataset. $p(w_i|s)$ is computed by the total # of the word w_i divided by the total # of words in the spam email of the training dataset. In case w_i does not appear in the training set, we add a smooth parameter $\alpha > 0$ to the probability such as $p(w_i|s) = \frac{\alpha}{total\ \#\ of\ words\ in\ spam + \alpha N}$ where N is the total # of unique words in the whole dataset.
4) Similarly, we need to compute $P(M|h) = p\left(w_1^{r_1}, w_2^{r_2}, \dots, w_n^{r_n}|h\right) = p(w_1|h)p(w_2|h)\cdots p(w_n|h)r_1 r_2 \cdots r_n$ where $w_i^{r_i}$ indicates that w_i appears in the email M r_i times and $p(w_i|h)$ is the probability of the word w_i in ham email of the training dataset. $p(w_i|s)$ is computed by the total # of the word w_i divided by the total # of words in the ham email of the training dataset. In case w_i does not appear in the training set, we add a smooth parameter $\alpha > 0$ to the probability such as $p(w_i|s) = \frac{\alpha}{total\ \#\ of\ words\ in\ spam + \alpha N}$ where N is the total # of unique words in the whole dataset.
5) Finally, we can compute $p(s|M) = p(M|s)p(s)$ and $p(h|M) = p(M|h)p(h)$. Since the numbers are really small, so compute log summation instead. $\log p(s|M) = \log p(M|s) + \log p(s)$ and $\log p(M|s) = \sum_i \log p(w_i|s) + \sum_i \log r_i$. Likewise, you may compute $\log p(h|M)$ accordingly.
6) Your prediction will be based on the larger log value.

Algorithm

1) Randomly split the dataset into 70% for training and 30% for testing.

2) Compute all quantities from the training dataset.
3) Create a function that accepts a row/record from the test dataset and return a pair of label and predicted label. This function will be used for the map on the test dataset.
4) Compute accuracy: $\frac{total\ \#\ of\ matched\ label\ in\ 3}{total\ \#\ of\ test\ records}$

Implementation Hints in Scala:

a) To collect all email messages, use reduce function on RDDs.
b) To get unique words, use toSet() function to cover a collection to a set.
c) To associate a probability with a word, use Map data structure, similar to associate arrays.
d) To search on a collection, import scala.collection.Searching._ and user search() method.
e) To find out how many repeating words, sort a collection, search and count. You may also use groupBy() method on a collection.

What to submit?

Take screenshots of your program outputs that contain the validation accuracy, and turn it in with your Scala programs in D2L.