Hash Table - Spam Classifier

Date assigned: Monday, November 21, 2011
Date due: Tuesday, December 6, 2011, **9:30 pm** **NO LATE ASSIGNMENTS**
Points: 60

We have implemented the Hash Table and a Priority Queue. For this assignment you need to get your Priority Queue to work using the Dynamic List (most of you have already done this in Lab).

We want to build a *naive* **Bayesian Spam Classifier**. To do this, we must count how many spam messages contain a particular word and how many ham (non-spam) messages contain that same word. We can use that information to calculate the probability, given all the unique words a message contains, that the message is spam. The algorithm is explained on the slides provided in class.

With any *machine learning algorithm,* you must run your classifier over some training messages to learn which words are likely to show up in spam messages.

Learning

Read data from a training set of email messages that have been pre-classified as spam or ham. Count the number of messages that contain each word you see. This allows you to determine the probability that a word signifies a spam message. You will need to use at least two hash tables (each of size 997), one to track the number of spam messages a word appears in and the other to track the number of ham messages a word appears in.

Application

Read a new email message and produce a spam probability based on the words that message contains. Insert the new email message into a Priority Queue using the spam probability as the priority (probability * 100 to convert to an integer, remember 0 is high priority) to simulate a user having his or her email inbox sorted by spammy-ness.

I will provide you with three sets of files, spam, ham, and unknown. All the files are actual email messages taken from the dataset enron1 on the web site: http://www.aueb.gr/users/ion/data/enron-spam/ . All email messages are preprocessed so that you can reliably read words separated by whitespace. Note that this algorithm is not perfect. Some unknown messages will be mis-classified by the algorithm you implement.

The Algorithm

The algorithm is explained in detail on the Wikipedia page listed at the end of this document. I've worked through the details with you in class and posted the algorithm on the class web site.

Details

You must provide a driver, spamClassifierDriver.c that produces the executable **spamClassifierDriver**. The Makefile must produce this driver at the root of your project. You must produce a spam classifier module. Use the hash function and compare function defined in the previous Hash Table assignment.

1. Your code is to be written in C using Eclipse 3.7. Programs written in other environments will not be graded. Create an Eclipse project named cs300_spam_PUNetID. This project should contain the directories: src, include, bin, and data. The driver, spamClassifierDriver, should be created at the top level of the project, not in the bin directory.
2. The Makefile must contain the necessary targets to build the spamClassifierDriver as well as a clean and dist target similar to the identically named targets in your Stack assignment. There is no testMe target required for this assignment. Typing **make** on the command line should build spamClassifierDriver.
3. Submit a file called cs300_spam_PUNetID.tar.gz into the CS300 Drop Box by 9:30pm on the day in which the assignment is due. This file must contain your **Hash Table**, **Dynamic List**, **Priority Queue,** and **Spam** projects. Submit a color, double sided, stapled packet of code by that same deadline. The packet should be in the following order:
   Makefile
   spamClassifierDriver(.h then .c if you have both, otherwise just .c)
   spamClassifierModule (.h then .c)
   Any extra .h/.c pairs you have.
4. Test one function at a time. This will lessen your level of frustration greatly.
5. You are to use the coding guidelines from V6.0 of the coding standards.

Goals for this assignment:

1. Code and test your program one function at a time.
2. Write efficient/clean code
3. Use the debugger to effectively develop a correct solution
4. Thoroughly test your code.

Output:

Each filename is a unique integer. You'll be putting this integer into the Priority Queue. The priority is (probability of spam of a message * 100) [to produce an int priority].

Train on the **spam** and **ham** files. Process each **unknown** file, calculate each file's spam probability and insert the file into the Priority Queue.

Remove each item from the Priority Queue to print the following Email In-Box. Messages with a spam probability of over .3 (or 30 as an int) are SPAM :

```
012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
Filenumber | Spam Probability | SPAM or HAM
-------------------------------------------
      1006 |                0 |         HAM
      1015 |                0 |         HAM
      1025 |                0 |         HAM
<many lines deleted>
      2175 |               29 |         HAM
      3871 |               31 |        SPAM
<many lines deleted>
       924 |              100 |        SPAM
       941 |              100 |        SPAM
       963 |              100 |        SPAM

Spam Hash
KEY                           |       COUNT
-------------------------------------------
 <unprintable characters>     |           1
                    antivirus |          14
                    byronizes |           1
<many lines deleted>
                    worldexpo |           1
                    wyfmgrqgp |           1

Ham Hash
KEY                           |       COUNT
-------------------------------------------
                    antivirus |           1
                    community |           7
                    construct |           2
<many lines deleted>
                    rewritten |           4
                    visionary |           1

Spam Hash
Longest Chain: 136
Total Number of Elements: 35654
Average Access Time: 32.0395
Number of Empty Buckets: 8

Ham Hash
Longest Chain: 96
Total Number of Elements: 19157
Average Access Time: 20.3282
Number of Empty Buckets: 77
```

NO LATE ASSIGNMENTS

WARNING! THIS IS DATA FROM THE WILD!

There are around 54,000 unique "words" in the email messages. Some words are numbers. Some are nonsense, some are unprintable Unicode characters, some words contain the delete or backspace character which breaks your table formatting. Assume the max word length is 1024.

I have included a file key.txt that tells you whether an unknown file was originally marked as spam or ham.

Use `./spamClassifierDriver > spamResults.txt` to write the HUGE output to a file.

http://zeus.cs.pacificu.edu/chadd/cs300f11/Bayes.pdf

http://en.wikipedia.org/wiki/Bayesian_spam_filtering#Mathematical_foundation

I encourage you to bring me your spam classifier module design!

This assignment is based on:

https://docs.google.com/document/pub?id=1aqDryoF3uFzw2T_6AkXiWR_h8feXU4DN1tqAw_DIEKo&pli=1

From Jaime Spacco at Knox College.

This assignment took me roughly 400 non-comment lines of code to write.