Spam Filter

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

http://en.wikipedia.org/wiki/Bayes%27_theorem

Spam Filter

- What is the probability that a message is spam?
- What is the probability that a message is spam, *given the set of words in that message*?

Conditional Probability

What is the probability of A given B?

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Unconditional Joint Probability

given

Bayes Theorem

Link P(A|B) to P(B|A)
hopefully, one of those terms
is easy/possible to calculate.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

only if P(B) != 0

P(Word | Spam) P(Spam | Word)

Which can we calculate?

Training/Learning

- If we have a collection of spam and ham messages, we can calculate

  P(Word | Spam)

- Which is convient, because we want to know:

  P(Spam | Word)

Bayes Application

$$P(Rare|Pattern) = \frac{P(Pattern|Rare)\,P(Rare)}{P(Pattern|Rare)\,P(Rare) + P(Pattern|Common)\,P(Common)}$$

Rare = spam
Common = ham
Pattern = word

Spam probability from a given word

$$P(S|W) = \frac{P(W|S)\,P(S)}{P(W|S)\,P(S) + P(W|H)\,P(H)}$$

P(S | W) is the probability that a message is a spam, knowing that the word "X" is in it;
P(S) is the overall probability that any given message is spam;
P(W | S) is the probability that the word "X" appears in spam messages;
P(H) is the overall probability that any given message is not spam (is "ham");
P(W | H) is the probability that the word "X" appears in ham messages

Spam Prob from many words

$$p = \frac{p_1 p_2 \dots p_n}{p_1 p_2 \dots p_n + (1-p_1)(1-p_2)\dots(1-p_n)}$$

p is the probability that the suspect message is spam;
p1 is the probability p(S | W1) that it is a spam knowing it contains a first word ("X");
p2 is the probability p(S | W2) that it is a spam knowing it contains a second word ("Y");
etc...

This assumes all words are independent
- not really true
- ok assumption in practice
- Naive Bayes Classifier

p gets very small
underflow!

Practical Formula

$$\frac{1}{p}-1=\frac{(1-p_1)(1-p_2)\ldots(1-p_n)}{p_1 p_2 \ldots p_n}$$

$$\ln\left(\frac{1}{p}-1\right)=\sum_{i=1}^{N}\left[\ln(1-p_i)-\ln p_i\right]$$

Practical Formula

$$\eta = \sum_{i=1}^{N} \left[\ln\left(1 - p_i\right) - \ln p_i \right]$$

$$\frac{1}{p} - 1 = e^{\eta}$$

$$\rightarrow \quad p = \frac{1}{1 + e^{\eta}}$$

Implementation

percentSpam = # spam messages / (# spam messages + # ham messages)
percentHam = # ham messages / (# spam messages + # ham messages)

pS(w) = **#spam messages word *w* occurs in** / total number of spam messages
pH(w) = **#ham messages word *w* occurs in** / total number of ham messages

spamacity(w) = (pS(w) * percentSpam) / (pS(w) * percentSpam + pH(w) * percentHam)

for all words, w, in a message, m
    sum += log(1-spamacity(w)) - log(spamacity(w))

spam rating of a message = 1/(1+e^sum)
    assuming sum !=0

Technical Details

#include <math.h>

double log(double value); // ln value

double log10(double value); // $\log_{10}$ value

double exp(double value); // e ^ value

```
gcc -o spamClassifier ... bin/hashtable.o -lm
```

libm.so

Technical Details

#include <stdio.h>

int sprintf(char *str, const char *format, ...);

printf formatting printed to str rather than screen or file.

```
char str[100];
int  value = 1234;
sprintf(str, “%d”, value);
```

Technical Details

- Read many files from a directory:

```
#include <sys/types.h>
#include <dirent.h>

    //http://www.metalshell.com/source_code/116/Read_Directory.html
    DIR *pDir;
    struct dirent *psDirEntry;
    char fileName[100];

    // open directory
    if (NULL == (pDir = opendir ("data/spam")))
    {
        perror ("opendir");
    }
    // read each entry from the directory
    while (NULL != (psDirEntry = readdir (pDir)))
    {
        // skip any file name that starts with .
        // skip .  and  ..
        if (0 != memcmp (psDirEntry->d_name, ".", 1))
        {
            // read file, Do Work, etc
        }
    }
    closedir (pDir);
```

Not part of C Standard, part of **POSIX** standard

Available on many Unix-like OSes

Hints

- if p_x is 0 or 1, you'll probably get bad results
  - might fudge to 0.05 or 0.95
- Not all words appear in both spam and ham messages
- Only count a word once per message!
- Design, Design, Design.

You probably need more than two hash tables

Hints

- I will provide you the spam/ham/unknown files
- Install these files in the **data** directory in your Eclipse Project
  - Do NOT submit these files to Subversion
    - To speed up a commit
      - Right click individual files to commit.

odds & ends

- gcc will tell you what file a file depends on
  - Just local dependencies

    gcc -MM hashtable.c

    chadd@coffee:~/> **gcc -MM src/hashtable.c**
    hashtable.o: src/hashtable.c \
     src/../include/hashtable.h \
     src/../include/../../CS300DynamicList/include/list.h
  - All dependencies

    gcc -M hashtable.c
- http://mad-scientist.net/make/autodep.html
- valgrind -v --leak-check=yes –track-origins=yes ./driver

odds & ends

- Don't overflow your buffer during fscanf

```
char wordData[1024];

while (EOF != fscanf (pFile, "%1023s", wordData))
```

Read at most 1023 characters and then add a NULL

turn a string into an int:

```
#include <stdlib.h>

int x = atoi(“1234”);
    x = atoi(wordData);
```