Lab 2:Spam Filter

In this lab, you will implement part of a naive Bayes' spam classifier.

To illustrate how this filter works, consider the following email:

```
Hey!
This is the best link I found. I thought you would want to see it!

www.somelink.com/example

Best,
Sus
```

We want to classify this email as either `spam` or not `spam`. Typically, the filter will consider the entire email and look for multiple words that are common in spam emails. For our filter, we will consider a single word.

For this example, we will classify the email based on the word "best". Assume the probability that any particular email is spam is 0.25, and the probability that any particular email is not spam is 0.75.

To classify the mystery email (above), we want to compute the probability that this email is spam given that it contains the word "best". Then we want to compute the probability that this email is NOT spam given that it contains the word "best". We then classify based on which probability is higher.

Let's define a couple of variables:

1. **C**: email contains the word "best"
2. $\overline{C}$: email does NOT contain the word "best"
3. **S**: email is spam
4. $\overline{S}$: email is NOT spam

Hence, we want to compute $P(S|C)$ and $P(\overline{S}|C)$.

Computing the Probability of "best"

First, we need to figure out how common "best" is in spam emails and how common "best" is in emails that are not spam. To do this, we have to use sample emails. This is called training data.

For this example, we'll use the following emails.

These are the sample spam emails:

1.
   ```
   you've been selected as a winner!
   click now to get the best anti-virus scanner!
   ```

2.
   ```
   you're a winner!
   reply immediately to claim your access to the best weight loss system ever.
   ```

These are the sample not spam emails:

1.
   ```
   hi,
   thank you so much!  you're the best!
   ```

2.
   ```
   hi,
   are you going to be here on friday?
   ```

Since we have two categories, we can only compute conditional probabilities. Those probabilities are $P(C|S)$ and $P(C|\overline{S})$, the probability of "best" given a spam message and the probability of "best" given a non-spam email, respectively.

There are two spam emails with a total of 30 words. Of the 30 words in the spam messages, 2 of those words are "best". Hence, $P(C|S) = \frac{2}{30}$.

Similarly, there are two emails that are not spam with a total of 17 words. The word "best" only appears once in any of the non-spam emails. Hence, $P(C|\overline{S}) = \frac{1}{17}$

Using the Law of Total Probability,

$$
\begin{aligned}
P(C) &= P(C|S)P(S) + P(C|\overline{S})P(\overline{S}) \\
P(C) &= \frac{2}{30} * 0.25 + \frac{1}{17} * 0.75 \\
P(C) &= 0.0608
\end{aligned}
$$

Throwing Bayes' Theorem In

To recap, we want $P(S|C)$ and $P(\overline{S}|C)$.

We also have

- $P(S) = 0.25$
- $P(C|S) = \frac{2}{30} = 0.0667$
- $P(C|\overline{S}) = \frac{1}{17} = 0.0588$
- $P(C) = 0.0608$

Bayes' Theorem tells us that

$$
\begin{aligned}
P(S|C) &= \frac{P(C|S)P(S)}{P(C)} \\
&= \frac{0.0667 * 0.25}{0.0608} \\
&= 0.2743
\end{aligned}
$$

$$
\begin{aligned}
P(\overline{S}|C) &= \frac{P(C|\overline{S})P(\overline{S})}{P(C)} \\
&= \frac{0.0588 * 0.75}{0.0608} \\
&= 0.7253
\end{aligned}
$$

Thus, we would conclude that this email is not spam since 0.7253 is greater than 0.2743.

Program Specifications

1. Ask for a word contained in the mystery email.
2. Using the sample emails that are provided, compute and display the following probabilities:
   - the probability that the word occurs given the email is spam ($P(C|S)$)
   - the probability that the word occurs given the email is NOT spam ($P(C|\overline{S})$)
   - the probability that the mystery email is spam
   - the probability that the mystery email is NOT spam
3. Print out the email's classification - Spam or Not Spam

Submit a PDF containing a screen shot of your program running. **Use 'diet' as the word and the files in testFiles.**

Some Details

- The sample email files provided are in all lowercase. You don't have to account for differences in capitalization.
- Your program should work for any five files named email1.txt, email2.txt,... and spam1.txt, spam2.txt,...
- Assume the probability of spam mail is 0.25, and the probability an email is not spam is 0.75.
- You should remove the following punctuation to accurately count words: periods, commas, exclamation points (!), and question marks (?)
  You can leave any other punctuation alone.
- It is possible for a particular word to have a probability of zero. We don't want that. To help alleviate this issue, you should add one to all of the counts.

  For example, going back to the example above, the probabilities would be $\frac{3}{31}$ and $\frac{2}{18}$
- Don't worry about rounding. (But if it bothers you, round to 4 decimal places at the very end)

Hints and Tips

- If you want to format your output, check out the iomanip header.
- File IO is managed by the `fstream` header. You want to create an `ifstream` object. (http://www.cplusplus.com/reference/fstream/ifstream/)
  You can read in a file word by word by using the stream operator (exactly same way you use cin to get input from the user)
- You can erase characters from a string (http://www.cplusplus.com/reference/string/string/?kw=string)
- Breaking up your program into functions will make your life easier. You can have one that handles getting the word counts from a file, one that computes Bayes' Theorem, and even one that handles getting input.

Check list

Your code must have a name header as described in the syllabus. Each of these items is worth two points of your grade.

Your output must

- Have a title in the output
- Include a short paragraph indicating the purpose of the application to the user and any directions the user needs.
- Use complete and clear sentences – do not assume the user knows what you expect from him/her.
- Have NO grammatical errors and/or spelling errors.
- Have NO typos – this includes using capital letters at the beginning of a sentence and punctuation at the end of sentences.

Example Output

Here is an example for what your program's output could look like. Yours doesn't have to look exactly like this, as long as it prints out the right things. User input is in blue.

This example uses the files from `smallTestFiles`

```
--------------------------------
Title Goes Here
--------------------------------
Purpose Goes Here
--------------------------------

Which word is contained in the mystery email?  winner

Probability of a spam email containing the word winner:  0.0967742
Probability of a non-spam email containing the word winner:  0.0555556
Probability the email is spam:  0.367347
Probability the email is not spam:  0.632653

Your message is not spam!
```

This example uses the files from `testFiles`

```
--------------------------------
Title Goes Here
--------------------------------
Purpose Goes Here
--------------------------------
```

```
Which word is contained in the mystery email?  diet

Probability of a spam email containing the word diet:  0.00798176
Probability of a non-spam email containing the word diet:  0.00128535
Probability the email is spam:  0.67426
Probability the email is not spam:  0.32574

Your message is spam...
```

What to Turn In

Upload your .cpp, .h (if any), and pdf to Canvas. Double check you've used the correct input for the pdf!