|  |  |  |
| --- | --- | --- |
| Patrick Tan  [ptan@college.harvard.edu](mailto:ptan@college.harvard.edu) | Timothy Kang  [tkang01@college.harvard.edu](mailto:tkang01@college.harvard.edu) | Kevin Zhang  [kevinzhang@college.harvard.edu](mailto:kevinzhang@college.harvard.edu) |

CS51 Final Project: Text Classification

**Brief Overview**

Documents exist to express an opinion or share information on a subject. The opinion or information shared can take many forms, but generally are limited to several possibilities for any particular topic. For example, reviews of a movie are generally either positive or negative; blog posts can be humorous or serious; and emails can be work-related or personal. The project we are undertaking seeks to develop and apply an algorithm to classify text documents based on their content. Using the Naïve Bayes algorithm, our algorithm will allow us to “seed” a program with a set of documents along with their known characteristics, and create a program that is able to classify a document presented to it using the same characteristics used to seed the program.

**Extensions**

The first extension we can implement involves writing the program in a dynamic way; although we seed our program using an initial set of data, the program, without a dynamic implementation, will become a static entity that does not grow as it is used. We can write our code in such a way as to continually update the stored data as more and more documents are fed to the program. This approach requires that our initial algorithm be accurate enough to ensure accurate results (for the most part) with later documents. The Naïve Bayes algorithm has been shown to be fairly accurate after only 250 test documents, so this should not be a concern.

The second extension we can implement is integration with the Twitter API (and potentially Facebook’s or Instagram’s as well). Tweets are especially great for our project since they are usually very opinionated and contain a few number of words that makes it easy to analyze and use to train the program. Implementing this extension may be difficult, but only because of technical issues not compatability; pulling data from the Internet and filtering out nonsense may be the hardest challenge here.

The third extension we can implement is something called n-grams. N-grams have to do with strings of words instead of individual ones; in particular, the modification we will make to our program in order to include this extension is allow the program to evaluate strings of words instead of only one word. Runtime and space complexity increase exponentially with this extension; in addition, coding a program to store every possible string of words will be difficult. A larger seed will be needed as well.

**Methods**

We will need two main methods for the bulk of this project: the first will read in sets of documents along with their classifications (that we have determined manually) and create a database that the program will use to evaluate future documents that are fed to it. The other method will take in a document that we wish the program to evaluate using the Naïve Bayes algorithm. Steps beyond this will include the extensions that are mentioned above.

**Structures**

Documents will be abstracted into a class; one variable will be a large string containing the full text of the document. One of the class variables should be a method that returns a list of strings, where each string is a single token in the large string delimited by spaces.

**Steps**

We will begin with the two main methods. These will consist of many parts:

~~Method to accept a seed data set~~