Project Proposal - Dum Dum

Team Name: FriENDers

Members: Kyle Blackmon, Jarrod Sanders, Ranger Chenore
Project Name: Dum Dum

Problem:
Our proposal, an application called *Dum Dum*, attempts to solve the age-old question: *Which of your friends is the dumbest?* While this might not seem like a crucial problem, the current age of "fake news" has proven the need for validity checks across social media. In part, the problem we are attempting to solve is: *Are your friends and followers a credible source of information?*

Motivation:
Our motivation to solve this problem stems from our generation's hyper-awareness of internet scams, false information, and social media. Due to our increasing dependence on social media as a source of information, our generation needs to be able to acknowledge when anything on the internet is misleading.

Features:
Our goal is to make some sort of GUI or CLI where users can enter a twitter account handle, select several options for scans, and then get a list of related people and their "intelligence ratings." This "intelligence rating" is arbitrary and in no way can we accurately represent a person's intelligence level through our simplistic social media scans, but it serves as a proof of concept.

Data:
For our project, we will be using an online, public database through the service Kaggle.com. We have attached a URL to the database site. The data is a simple .csv file. The given word is in the first column, followed by the frequency of that word's occurrence. There are a total of 333,333 elements in this database and it was originally pulled from Google Web Trillion Word Corpus. [1]

Tools:
We will use Python to access the Twitter API and write data to a CSV file, then we will use C++ to perform the intelligence computations on the data written in the CSV. C++ will also be used on the kaggle data to create data structures that the code will utilize for analysis of the CSV data.

Visuals:
Our system will likely be controlled via a CLI, and will have minimal graphics, if any. We may expand the scope of the project to offer a GUI, but that is a stretch goal

Strategy:
We are currently planning to use a balanced BST to store the wordlist and weights, alongside a Trie to check if the wordlist contains the word in question to save time when processing misspellings/jargon. We will also use a hashmap for the same purpose; the key will be the string and the value at the key will be it's frequency rating according to Kaggle.com. We will then compare the performance of our program using each data structure.

Distribution of Roles:

Jarrod	**Write the C++ code containing the data structures/algorithms and processing.**
Kyle	**Write most documentation. Help with C++ coding and wherever else needed.**
Ranger	**Use Python to access Twitter API and generate CSV from data. Help with C++ coding to process that data.**

References:

[https://www.kaggle.com/rtatman/english-word-frequency**]**

[https://en.wikipedia.org/wiki/Trie**]**

[https://developer.twitter.com/en/docs/twitter-api**]**

Sketch of CLI / Program layout:

Page 3. (From the notes: 2 page limit excludes wireframes/sketches and references)

```
What operation do you want?
[1] get a single account score
[2] get followers scores
[3] get followed score
1
Enter a Twitter Account URL or @<name> (enter 0 to go up a menu):
@PresidentFuchs
-> @PresidentFuchs's score: 9001
Enter a Twitter Account URL or @<name> (enter 0 to go up a menu):
0
What operation do you want?
[1] get a single account score
[2] get followers scores
[3] get followed score
3
Enter a Twitter Account URL or @<name> (enter 0 to go up a menu):
@PresidentFuchs
What do you want to know (enter 0 to go to the menu):
[1] Top scoring followed
[2] Bottom scoring followed
[3] Average score of followed
3
-> Average Score of followed is: 420
What do you want to know (enter 0 to go up a menu):
[1] Top scoring followed
[2] Bottom scoring followed
[3] Average score of followed
Ctrl + C
...
```