William Wu, Fabio François, Ada Chen, Brandon Troche
Felix Grezes
June 21, 2016

Profil'd

I. Research Idea

Our vision is to be able to compare a user to a supported Twitter account. This will provide us a metric as to how someone possibly is in nature. We will start by analyzing the tweets of current presidential candidates and ISIS supporters. This selection will give us a good idea of who is more right wing and who is more left wing as well as how much of a terrorist they may seem. While this metric is probably not accurate enough to accuse anyone of any wrongdoing, given time and more sources to pull our data from, we will be able to compare people more accurately.

II. Tools & MEAN Stack

We will begin by using Python and the Tweepy library to grab tweets from the user's Twitter profile and storing them as JSON into MongoDB. Our MEAN stack will be the core of our web application. MEAN stands for MongoDB, Express, AngularJS and NodeJS. MongoDB as we discussed earlier will be our database, Express will be our skeleton for our webpage, AngularJS will be used to dynamically display our information and NodeJS will handle all the running of functions. When the user enters their Twitter handle and hits submit, we will run our Python script which can allow us to grab their page and their tweets, add them to our database, and then run Spark to compute tf-idf and similarity for each of our provided Twitter profiles. All of this will be hosted on Heroku.

III. Analyzing Data

In order to make the most accurate comparison between the user and the presidential candidates and ISIS supporters, we will look specifically at keywords used by both sides. We will identify the keywords by determining the most commonly used words by the politicians and ISIS supporters and then categorize them into different groups. To collect the data we will use Python's tweepy library to interact with the twitter API. Using tweepy we can collect up to 3000 of the user's most recent public tweets. We will compare these tweets to the keywords in

the database and should be able to represent a user's political preferences based on what they match up with.

IV. Web App

Our web app will allow users to input their Twitter handles for processing. Upon hitting submit, our function will run which will get their Tweets, store it in our database, then submit our script to Spark to get the tf-idf for each term and find similarity between these users and finally display the results in an aesthetically pleasing manner.

V. iOS App

The iOS app will be similar to the web app. We will potentially use Firebase to store the tweets instead. It will be more challenging as it will most likely be based on Swift and calling Python scripts in Swift will be difficult as there are no good libraries for it. Alternatively, we can implement the algorithms in Swift for the app so the app can function seamlessly in order to not allow this limitation on calling python hold us back. Since swift is a language that has a lot of versatility, mirroring the algorithms in it should not be too difficult. If all else fails, we will call a python script through swift from objective-c using a bridge between swift and obj-c.

VI. Goals

Our aim of this project is to provide an entertaining means of comparing tweets from the user to tweets from different personalities- in our case, politicians and ISIS supporters. It can possibly provide ways to realize threats when compared to supported twitter pages.

VII. Project Timeline

Our first goal for the group is to learn the technologies involved. Ada will be working on the front end, Fabio and William will handle the backend and perhaps Brandon can start on the iOS front end. After learning the technologies, we need to design our front end for each platform. After that, hopefully the backend functions can be finished and the database will be fully loaded with the necessary tweets. Upon hitting the submit button, at this point, it should be able to at least retrieve information from our database as a test. After that, ç the front end developers should format the data for the results page so that they display in a comprehensible manner. The backend developers at this point will try to call Spark from the same Python script that was produced earlier in order to obtain the tf-idf for each term and then compute the similarity between the pages. If this portion of the project is finished, we have reached our MVP.