CS 51 Final Project – Final Specification

ArOund

The Team

Andrew Malek
amalek@college.harvard.edu
Ivan Cisneros
ivanalexiscisneros@college.harvard.edu
Marc Abousleiman
marcabousleiman@college.harvard.edu

TF: Winnie Wu

Signatures/Interfaces

Crawler

- Primarily, uses tweepy api to send GET requests from certain twitter accounts.
    - These sources accounts are predefined in a separate text file via their account id.
- Contains function that sends GET requests to twitter accounts in the sources file.
- Contains method for filtering out unnecessary information provided in the request, and keeping only the content, author, tweet id, and tweet location.
- Outputs this data as a csv file, which will then be iterated through by graph module.

Graph

- Uses input tweets from crawler module.
- Contains a hardcoded balanced binary tree of cities; each node will contain the tuple (city_name, (lat, long), pointer_to_event_tree_in this city)
- Uses a predefined, hardcoded dict of keywords which are matched to the crawled tweets; each match adds the event to a binary tree of similar events.

- A pointer from the "cities" tree points to the relevant "events" tree; this method will be used to relate location of event to the events themselves.
- Contains function that intersects trees for the same event, which is found through a "similarity score" that uses keywords in the tweet to verify that they are the same events.

PageRank

- Uses input graphs from graph module.
- Contains function to iterate through "cities" tree to find the relevant cities (which is restricted by the area of the map that is being viewed on the app interface).
- Contains function that establishes an "importance" score for the events in the corresponding events tree of that city.
    - Uses predefined metrics (legitimacy of source of the tweet, number of tweets that were used to establish this events object – essentially the similarity score found in graph -, gravity of event) rather than traditional markovian chains.
- Contains function to rank events based on the importance score and feed this rank to the app GUI.

Timeline

Week of April 20th – 26th

- Finish work on graph, crawler, and pagerank modules so that that they work in sync.
- Begin work on the iOS app UI.
- Figure out how to limit city requests through the app so that pagerank knows which cities to limit its ranking for.
- Upload relevant backend code to the server.

Week of April 27 – May 2nd

- Finalize work on iOS UI.
- Finalize implementation of backend python modules if any bugs or inconsistencies are found.

- Allow for iOS app to communicate with server and send data requests.
- Implement events/tweet display on the MKMapView.

Progress Report

Activities completed since last report:

- Set up a repository system using GitHub (repo origin: https://github.com/andrewmalek27/arOund.git).
- Set up our framework and standardized our workflow.
    - Made sure all team members had the same version of Python.
    - Downloaded and familiarized ourselves with the Twitter API framework that we chose ("Tweepy", found at https://github.com/tweepy/tweepy).
    - Downloaded GitHub GUI and familiarized ourselves with operating.
- Met with our TF to discuss the details of the project.
- Began work on the Twitter crawler module; currently outputs the necessary data as csv files.
- Began work on the Graph module and the event object type; established main graph data type to be balanced binary trees.
- Set up a Digital Ocean Server (installed with the same version of Python and Tweepy that we are using on our local machines), from which the backend data crawling/matching will be done.