Code Documentation

# Pre-requisites:

* Python 2.7 or higher
* Neo4j Desktop 3.5.6
* Node current version
* Twitter API token, visit: <https://developer.twitter.com/>
* YouTube API token, visit <https://developers.google.com/youtube/v3>

# Importing database:

* Install Neo4j Desktop from the link
  + <https://neo4j.com/download/>
* Open Neo4j application and create new graph named 'Social Search' but don't run it.
* Click on 'Manage' and then click on 'Open Folder'
* From the File Explorer, go to 'bin' folder and open command prompt
* Run the following command

neo4j-admin load --from=<LOCATION OF **graph.db.dump** FILE> --database=graph.db –force

* Now, in the Neo4j application, start the newly created graph database

# Installing and setting up python virtual environment:

* Go to your command prompt and type
  + pip install virtualenv
* Navigate to the project directory
  + cd socialsearcher
* Create a virtual environment using the command
  + virtualenv env
* Activate the virtual environment by typing the command
  + venv\Scripts\activate
* Deactivate using the command
  + deactivate env

# Installing python dependencies

* Navigate to the project directory using command prompt
  + cd socialsearcher
* Activate the virtual environment using the command
  + venv\Scripts\activate
* Run this command to install dependencies
  + pip install -r requirements.txt --no-cache-dir
* Run this command to install preprocessor module
  + if *git* is installed on system:
    - pip install git+https://github.com/s/preprocessor.git
  + if *git* is not installed on system:
    - pip install --upgrade <https://github.com/s/preprocessor/tarball/master>
* Input the Twitter API by replacing the placeholder strings in *api.py*
* Run the API from command prompt using
  + python api.py

# User Interface

* Install node from the following link
  + <https://nodejs.org/en/download/current/>
* Go to the 'UI' folder and run this command using the command prompt to install node modules
  + npm install
* Now, run this command to launch the project
  + npm start

**API Documentation**

**GET** http://localhost:8000/api/socialsearch/tweet

**Description**: GET endpoint for extracting tweets of a specific story

**Parameters**:

* id: the id associated with each story

**Returns** a JSON object consisting of tweets

**GET** http://localhost:8000/api/socialsearch/video

**Description**: GET endpoint for extracting videos of a specific story

**Parameters**:

* id: the id associated with each story

**Returns** a JSON object consisting of videos

**GET** http://localhost:8000/api/socialsearch/entity

**Description**: GET endpoint for extracting all entities connected to a specific story

**Parameters**:

* id: the id associated with each story

**Returns** a JSON object consisting of entities

**GET** http://localhost:8000/api/socialsearch/summary

**Description**: GET endpoint for extracting summaries of keyword/story

**Parameters**:

* id: the id of the user that initiates query
* key: the search query

**Returns** a JSON object consisting of stories and/or sub-stories

**POST** http://localhost:8000/api/socialsearch/login

**Description**: POST endpoint for user login authentication

**Parameters**:

* email: user's email for login
* password: password associated with email

**Returns**:

* if, credentials are valid, a JSON object
* else, a string showing an error

**POST** http://localhost:8000/api/socialsearch/register

**Description**: POST endpoint for registering user account

**Parameters**:

* email, should be unique
* name
* password

**Returns** a string indicating success or failure

**GET** http://localhost:8000/api/socialsearch/history

**Description**: GET endpoint for extracting user search history

**Parameters**:

* id: the id of the user

**Returns** a JSON object consisting of a list of searched terms

**POST** http://localhost:8000/api/socialsearch/edit

**Description**: POST endpoint for modifying user account

**Parameters**:

* id (mandatory)
* name (optional)
* password (optional)

**Returns** a JSON object consisting of user account data

**GET** http://localhost:8000/api/socialsearch/trends

**Description**: GET endpoint for extracting current trends

**Parameters**:

* none

**Returns** a JSON object consisting of top trends

**Scripts**

# twitter\_stream.py:

* **Description:** The data from twitter is fetched based on a keyword search using the standard Twitter API
* **Input:** Accepts a keyword that is used to fetch tweets from the Twitter API
* **Output:** Returns a JSON object of tweets, and dumped into a location

# youtube\_stream.py:

* **Description:** The metadata of videos from YouTube is fetched based on a keyword search using the standard YouTube API
* **Input:** Accepts a keyword that is used to fetch video metadata from the YouTube API
* **Output:** Returns a JSON object of YouTube videos’ metadata, and dumped into a location

# topic\_clustering.py:

* **Description:** The script uses the twitter data and clustered them using Silhouette analysis and LDA topic modelling, and identifies topics among the tweets and videos
* **Input:** Accepts a keyword on which tweets and videos are fetched
* **Output:** Returns a JSON object of tweets and videos with topics identified, and dumped into a location

# text\_summarization.py:

* **Description:** The scripts uses the data from each individual cluster and generates a summary of the top tweets
* **Input:** Accepts a keyword that on which tweets are fetched
* **Output:** Returns a JSON object of summaries, and dumped into a location

# create\_graph.py:

* **Description:** The script fetches all the twitter, video, and summary files and creates a graph with entities and relationships
* **Input:** Accepts a keyword that on which each JSON files is fetched
* **Output:** Creates a Neo4j database

# api.py:

* **Description:** The API that communicates the client application with the neo4j database. The data is retrieved using the *py2neo* client of python which is used to run **Cypher** queries.
* **Input:** none
* **Output:** Serves different endpoints for accessing data

**UI Modules**

# App.js:

* **Description:** This module contains different routes like Timeline, History, About, Profile etc.
* **Input:** endpoints
* **Output:** Serves different end points to access UI pages.

# Timeline.js:

* **Description:** This module renders the home page of the system contain sub modules like Trends (Sidebar) which display some top twitter trends, Navigation Bar to navigate to different pages, Search Bar for searching user query and Timeline (summary) to display summaries of given keyword in a timeline fashion.
* **Input:** trends, summaries
* **Output:** Timeline of summaries

# SignupLogin.js:

* **Description:** This module renders the Signup and Login screens.
* **Input:** Credentials, User Information.
* **Output:** Home page

# Profile.js:

* **Description:** This module renders current user profile; user can also update their information.
* **Input:** user information (if edit)
* **Output:** User profile

# Logout.js:

* **Description:** This module deletes current user session
* **Input:** none
* **Output:** Login/Signup page

# History.js:

* **Description:** This module renders the user search history in a tabular form user can also search from there by clicking the search button
* **Input:** none
* **Output:** user search history

# About.js:

* **Description:** This module renders the developer information
* **Input:** none
* **Output:** developer emails and information.

# TweetsTimeline.js:

* **Description:** This module is a sub module of Timeline.js it renders tweets and YouTube videos of specific topic, summary of these tweets also given on the top. User can also sort tweets by different options available in a drop down. This module also contains a sub module named Entity which renders the entities related to these topics.
* **Input:** Topic id
* **Output:** Tweets and videos of the give topic id