### 2.1.1

**Issues:**

The task of this part is getting relevant messages through Twitter according to queries that users input. Based on the requirement, it can limit the query to a specific geographic region(latitude/longitude). Moreover, a demonstration of people who retweeted each message must be achieved. And the most difficult thing is to select tweets that are really poplar retweeted by other users and contains the keyword and combine these query requirements with location limitation.

**Design and its Motivations:**

First part we use Tweeter ‘Search/Tweets’ api to get useful information according to user’s input. According to the requirements, we should use keywords and geographic region(like latitude and longitude) to get discussions. Firstly, we have to get users’ key word query and optional region data. This program use html5 get geolocation function which can get user’s location when it is allowed by user. As for keywords, the form can handle texts which contains useless words such as ‘a’, ‘the’ and others. This maintains the keywords are really popular and meaningful. Then twitter ’Search/Tweets’ api provides three useful parameters which are query, coordinates and result\_type which specify the requirements and return correct information. When parsing the return Json format data of the request, there are users, texts and relevant information can be stored in an array and shown on the web interface.

Second part is providing a hyperlink which helps users to check who have retweeted the selected popular tweet. We select ‘Retweet/:id’ API function which use tweet’s unique id to query who retweeted the information and relevant information. The tweet’s id was stored in the first part querying topics and the response information also provide retweets’s author profile url and author name. All these information helps make the result and web interface more beautiful.

Finally, the advantages of this solution works quite efficiently with tweeter API and comply with the requirements well. But the disadvantages of that are program can not get the latest tweets which were created no more than a hour or less and didn’t handle the results with hash tag and detect the popularity of the tweets.

**Requirements:**

We can fully satisfy the requirements, which are showing the relevant information and displaying the users who retweeted the messages. And below is the requirements form:

|  |  |
| --- | --- |
| Requirements | Complete degree |
| Input keywords and handle input texts | Fully complete |
| Limit query to specific geographic region | complete |
| Output message | complete and in right format |
| Display who retweeted message | Fully complete with user profile |

**Limitations:**

According to the resolution, the program can’t get latest popular topics which may resulted that if theres an emergency like earthquake or fire, the program may not get results immediately.

### 2.1.2c

**Issues:**

This part is to get a list of user by querying a venue’s name and check who have visited this venue in a limit days. If user input last X days text is empty, the program should using twitter streaming api to capture information when it is published. There are two main challenges of this section. First one is to select between twitter ‘Rest api’ and twitter ‘Streaming api’ according to users’ input. Second challenge is to query and store venue data and push checked in users’ in the venue array.

**Design and its Motivations:**

The design of this part use two APIs to get the correct output, first is using foursquare venue search API to get the venue’s coordinate and then use tweeter search API to get the users by querying their check in information. In order to get the correct result - a user list that appeared in a venue in the past few days, the first part is to get the correct location coordinates using Foursquare search api combined with location keywords parameter. The parameter is what user input in the form and handled by script functions that ensure user’s input is in right format. After getting a list of venues which contains venue keywords or related words, the server can compare the venue name with user’s input and return a real venue location information. Then the server use tweeter api to search the user list. In this part, if users’ input days are empty, the server could use tweeter Streaming API to search users and if days are integer, the server will use tweeter REST api. All returned data will be stored in a venueAndUser array and can be shown on the interface after well styled both on the map and a user list. In this part, user can index a selected checked in user’s past 100 tweets using Tweeter Timeline:id api.

The advantage of this design choice is that server can get correct venue information and checked in user list based on that information. The usage of ‘Streamging api’ and ‘Rest api’ make the user list keeps updating in real time. The disadvantage of this design is the venue might not be quite accurately shown on the map due to some unfixed bugs.

**Requirements:**

This solution partly fulfilled the requirement. All complete degree is shown below:

|  |  |
| --- | --- |
| Requirements | Complete degree |
| Input venues and days and handle input texts | Fully complete |
| Use Twitter streaming api and rest api | Partly Complete with few bugs using streaming api |
| Display user name, id, etc. | Complete |
| Display user’s 100 tweets | Complete |
| Information about users current whereabout | Partly complete with showing user’s location |
| Venues information List and show maps | Complete |

**Limitations:**

When users’ input keywords(such as Shop, Library, Museum etc.) are too normal to search in Foursquare, the results shown on the map are not so accurately. And users list will be too large because the server will add all users who have checked in all venues.

## 2.2 Querying the Web of Data

**Issues:**

The aim of this section is to find information about the surroundings of a venues visited by a person. To achieve function button “show point of interest”, we have to use DBPedia and Foursquare to query the information. The challenge of this part is to get correct information using SPARQL searching language.

**Design and its Motivations:**

The design of this part are searching recommended data from foursquare and DBPedia website. The difference between these two part are querying methods. When searching point of interest venues using Foursquare api, venue/search api with intent and radius parameters will be used. As for DBPedia, we use SPARQL searching language to search recommended venues’ latitude and longitude and then show these locations on the map. All venue locations will be shown on the map and popping up markers on the map. When clicking the markers the page will jump to foursquare related venue page. When press DBPedia point of interest button, it will jump to DBPedia recommended venue page.

As for the disadvantage of this design, if server cannot get a url link from DBPedia querying result, the page will not jump.

**Requirements:**

All requirements complete degree are shown below:

|  |  |
| --- | --- |
| Requirements | Complete Degree |
| Retrieve data from DBPedia and Foursquare | Complete |
| Output can be plotted on a map | Complete |
| Show information by opening its related webpage | Complete (Some venue information may can not get from [DBPedia.org](http://DBPedia.org) because of venue name not in wiki database) |

**Limitations:**

All query request and returned data is not in real time.

## 2.5 Web Interface

**Issues:**

For this part, we need to design an webpage for users to send their requests to the server. Because the requirement of this program is divided into three individual part for each teammate, the biggest problem we faced is that associating different parts to the final webpage we designed.

**Design and its Motivations:**

The web interface directly influent user’s experience. To ensure all the web requests shown on the page and transfer user’s input correctly, the web interface should maintain its webpage style beautiful and stable. Multiple CSS styles are used in this project and all webpages are created by using .ejs file, which is a page format combines data and a template to produce HTML.

**Requirements:**

|  |  |
| --- | --- |
| Requirements | Complete degree |
| System implemented as HTML/Javascript sets | Fully complete |
| System controlled through browser-based interface | Fully complete |
| Interface served by node.js server and all interaction with social web done through node.js server | Complete but server name is app.js |

**Limitations:**

# 