Recording of meetings with Prof Nick Avis & additional project notes

Week 1 (04/11/12) – Discussed

First meeting with Nick Avis, discussed basis of project and what information I gathered over the summer, including my previous contact with Omer Rana and Nick Horne.

Discussed direction for project – Initially try to complete a 'proof of concept' with an easily verifiable example to show that the current data collection and analysis tools work. Examples included football game or current news story.

As project progresses, branch out into other more abstract situations, using human intelligence to look for patterns to show that the system works, and ultimately to try and find the limitations of this analysis method.

Use predetermined events to work with – examples being US Presidential Election in November. Additionally, make use of real-time situations, such as **severe weather** or local/national news stories.

Also **focus on combining data from multiple sources**.

Task: Research Weather & News APIs, for use integrating into current system.

Decided to create new program instead of adding new features to old one. New program will however be dependent on the old one, i.e. taking long and lat's of tweets by accessing (and updating) the mongo database etc.

Ideas

Either: separate program, querying database for long and lats of all recorded tweets. Should be ran after collection program has ran. This will then flag up any tweets of interest for human inspection.

Week 1 - Achieved

Researched and studied existing Java code for tweet harvesting, mongoDB database, and a number of weather API's.

Initially began to develop code for the Metoffice DataPoint APi (datapoint.metoffice.gov.uk) API – met difficulties as I wanted to use longitudes and latitudes, but after contacting a developer there, they stated their API did not support this, and to implement my required functionality I would need to download 5000 forecast locations, and then perform the haversine formula

$$= 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Considered this, but I eventually decided on using a combination of API's from wunderground.com, and www.meteoalarm.eu.

Whilst there have been a number of complications , this was definitely the better choice, as not only does it provide me with an alerts system, but crucially the project can now detect weather from across Europe, which the Metoffice API could not do. Metoffice used meteoalarm for their weather alert ratings across the UK too.

Decided to create a new program, rather than modify existing code (however a lot of existing code had to be modified too, including how the system manages the database.) System currently:

- Inserts into database additional Boolean value depending on if each tweet has a geolocation – longitude & latitude
- Database is queried and all tweets with 'true' are extracted.
- Each tweet is then individually queried with the Wunderground API, which returns the current weather conditions based on the geolocation (i.e. RAF Brize Norton)
- The nearest available weather station is then obtained by parsing the XML document using DOM queries, via a URL.
- An alert query is sent via meteoalarm with the nearest weather station, to return any localised alerts (i.e. Flooding, Snow etc), along with a severity rating.
- These results, along with the respective Tweet ID number, are then printed into a text document

Tasks to complete

Primary task -Currently, the tweets are collected, and then I run my program to query the data etc. and get info. I would like to query this data in real time, so I would have both running at the same time. This then leads to opportunities such as automatically displaying tweets discussing the weather, from areas that have had an alert status issued.

Currently the system only works in Europe due to API limitations. I would like to make the system work worldwide.

Reduce as far as possible the number of API calls needed – due to a max quota per day and per minute(!)

Copy xml document from online to local storage, and check for geolocation info of town and weather station. Then pass this back into getstuff.java as variables, and to query /alerts/

Combined two queries – forecast and geolookup, into one to half time needed to request document + API requests.

Using meteoalarm for Alerts – "The information is supplied by over 20 of Europe's leading national weather services, including the Met Office"

Source: http://www.metoffice.gov.uk/weather/europe/meteoalarm/

Tasks:

- Get population of area
- Display map of area

Week 2 (04/11/12) - Discussed

Encouraged to continue focussing on weather aspects – a question such as 'Is the stereotype that the British discuss the Weather more than other countries justified?' was suggested.

No issues discussed – struggling with correct DOM query of XML document, not major issue.

Ensure time is left over for user testing and feedback at the end of the project.

Week 2 – Achieved

- Major code cleaning – printing only useful values + commenting all previous code. Solved DOM problems.

- Getting live info, reformatting data output into new 'refined' text document
- Created a GUI – When button pressed, all tweets not yet analysed are queried, and are checked to see if they pass the specified requirements (i.e. if they contain geolocation metadata, and are in an acceptable format to be sent to both APIs). Only these tweets are then sent queried – this is to reduce the number of API calls, due to the overhead incurred for each query + the limited number of API calls.
- After all tweets have been analysed, any useful data is sent to a 'refined.txt' document, and all other information is destroyed. The twitter-collection program can run simultaneously – each time the button is pressed, the set of data not yet queried is moved and dealt with, so the collection program can continue to input data into the original, but now empty, database.
- Extensive testing of software so far. Noted that only between 5 – 10% of all tweets have an associated **accurate** geolocation. Researched and considered using data such as IP addresses of the user to determine an approximate location, but research from DNS Stuff suggests these are only accurate around 50% of the time – other times giving incorrect locations. This incorrect data would harm the project more than it would help it, thus I will just use the accurate GPS data.
- To consider: some cameras now have a 'geo-tagging' feature – would it be possible to search for photos uploaded with this meta data, and harvest these too?

References:

http://whatismyipaddress.com/geolocation-accuracy

Tasks:

- Put a counter into GUI of how many tweets can be analysed – possible??
- Pop up menu if tweets from alert area found
- Colour alert box in line with alert level – yellow/orange/red

Week 3

Achieved:

- Major bug fixing and code cleaning.
- Got notification popups working correctly

Week 4:

Achieved:

Got satellite imagery & nearby webcam services working correctly for each new popup. Reformed pop up boxes to only display basic info, and have an option for more info if a user clicks a button – processing for extra information only occurs after this button has been pressed, leading to a more efficient overall program

Initially began by getting news originating from the tweets geolocation, using Globedia API - however, this news is in the native language so not particularly useful.

Now, I get any news from that day from a number of news sources, which has the name of the town/area in. Display top three stories.

Help Reference: https://gist.github.com/2320294

Week 5:

Task:

- **Have Flikr search and maps for each location working**

Things I can find with Yahoo Developer API:

Upcoming events – Description, Time + Place

US Election Harvesting

Recorded Ohio from 1:40am to 2:00am

Recorded key words from 2:05am

Obama

Democrat

Republican

Romney

Searched for keywords from 02:20am:

Election

Elections

College

Vote

Votes

Timeline – searched north south and central Ohio from 02:35am – 03:54am for semantic analysis

Week 6:

Transferred code to new computer – LOTS of code tidying + optimization in process.

Got in contact with wunderground.com and discussed my project – agreed to give me a full API key worth $300 per month, until July 2012 – a total value of $2700. Have to reference wunderground.com in interim report. Limits increased to 5000 calls per day & 100 calls per minute, with access to all features – referred to as Drizzle Anvil plan.

Discussed journal at infoviz 2012

Changed all maps from Google maps to Bing maps – this allows me to harvest real-time traffic data and overlay on top of maps. Additionally, there are less legal requirements with Bing maps – Google static maps can only be displayed within a web browser if used commercially. – 18/11/12

21/11/12

Have added extensive useful information to start-up GUI, and a number of checks to ensure the database is running correctly.