GTBT Report 28/01/2015

**Architecture**

The architecture of the project has changed since last year. The project previously comprised of three components with data passing between the three components using HTTP Requests.

This was changed so that the client application was more responsive and to reduce loading times as the number of AJAX calls needed to render a view was growing and affecting the usability of the interface.

Another reason for the change was that the client application was using the Django framework without any real requirement to do so. The CA only rendered the HTML/CSS and JS therefore the only purpose of the Django framework was to provide a webserver.

Therefore, the service component and the CA component were integrated. This integration should also make things easier to manage.

The current architecture is now 2 applications that share a common database (MongoDB).

**Feed-Enrichment**

Different document features and applying different pre-processing measures – such as removing URLs – has been implemented in the NLTK implementation. The results of this have been recorded and have shown improvements and are now included in the component.

The experimentation of document features and the results of each experiment have been included in the appendix.

More crowdsourcing was carried out and using the CrowdFlower platform I was able to add approximately 2000 tweets to the training set. I did attempt to carry out more crowdsourcing jobs however the cost became prohibitive as the cost estimation grew and the time to complete increased.

While working on the project between the 25th of December and the 1st of January, I accidentally deleted my database. I managed to retrieve approx. 50,000 tweets back from an earlier back up but I have had to start mostly from scratch. The database now contains approx. 300,000 tweets. I have also taken steps to stop a repeat of this happening by taking weekly backups.

I also created an implementation in Java using Weka. I started this because the resources need to build a model with NLTK exceeded the number of resources available in my MacBook. Another reason for the use of Weka was to reduce the time taken to build models. Weka also has a wider range of facilities for evaluating classifier performance.

The Weka implementation exploits the same features as the python version and pre-processes the tweets in a similar fashion albeit with a different stop-word list. Using 10-fold cross validation the accuracy of the implementation was ~70%

**Client Application**

The client application has included aesthetic changes to make it more appealing and easier to use (still to be proven!) see appendix for screen shots.

Some features implemented are:

* The user can now ‘pin’ individual tweets so that they can save them for viewing later. The ability to unpin is also implemented.
* Different Alert types have been added. The alert types are currently “New Discussion” and “Influential Author”, “New Topic”
* The user can now restrict the twitter data used by time e.g. past 24 hours, past 7 days, past month, etc.

The last feature listed has brought to light a few issues on how the tweets are stored in the database. The tweets timestamp attribute, “created\_at”, is stored in the database as a String when it should be a date. This meant when the user tried to filter by time the application was doing a string comparison rather than a date comparison. This problem is still present and I’m currently trying to work on a solution. This is a reason why the time-series graphs are not being displayed.

The list of all entities in the database has been removed from the UI. I figured this was unnecessary for an actual user to see and most likely would make it difficult for them to use. Instead, the user can add new entities to the system to track or click on the discover link which will show the entire list of entities.

I also attempted clustering of tweets to see if it were a possible means of condensing the data displayed on screen. I implemented this using single pass clustering using a threshold of 0.6. This did make the viewing of the information easier, but I had no means of evaluating the accuracy. The application currently displays these as “Topics” but there are no clusters in the database at present as it was an experiment that has not been implemented fully in the system.

**Appendix**

Testing was conducted using the collection of 2048 tweets provided . The collection was split so that 75% of the collection was used for training and 25% was used for determining the accuracy. The accuracy was calculated using the function provided by the NLTK library.

first method - 0.46875

use text terms as features,

tokenise and stem

second method - 0.470703125

use text terms as features

tokenise and stem

remove urls

third method - 0.53125

use text terms as features

use retweets as features

tokenise and stem

remove urls

fourth method - 0.537109375

use text terms as features

use favourites as features

tokenise and stem

remove urls

fifth method- 0.5390625

use text terms as features

use favourites as features

use retweets as features

tokenise and stem

remove urls

sixth method - 0.544921875

use text terms as features

use favourites as features

use retweets as features

tokenise and stem

remove urls

remove all tokens with punctuation