COM3504 Intelligent Web Report

Introduction

The aim of this assignment was to build a program able to query the social Web through the available API's to track people's movements and the topic of their discussions [1]. For this assignment football was the chosen topic.

Our solution makes use of twitter's SEARCH API to allow users to write queries that can be built up in many different ways, users may search by the following terms: team, players, hashtags or keywords [2]. Two buttons on the interface then allow the user to choose if they would like to search twitter for tweets containing ALL of these terms (an AND query) or ANY of these terms (an OR query).

If successful, tweets will be retrieved and displayed in a retrieved tweets pane on the screen. Each tweet is displayed in a format very similar to that of twitter. In addition to the tweets retrieved pane, a statistics pane is displayed, this contains a display for the top 20 words and 10 most active users along with their most used words for the retrieved tweets. Finally a google map centred on the UK is displayed which displays all geo located tweets if any have been found.

Our interface is implemented using HTML and Javascript and is served by a node.js server [3]. Data is stored in a MySQL database. All interaction with the social web is carried out through this server.

At this stage no additional features have been implemented.

1.1 Querying the Social Web

1.1.1 Issues

The task for this section was to query the social web. Relevant tweets should be retrieved with some statistics calculated and their locations displayed on a map (if applicable). Some of the main challenges faced when working on this section were: calculating statistic information efficiently, making sure an appropriate number of tweets were retrieved and using a database where appropriate to reduce the amount of calls to twitter [4].

1.1.2 Design Choices

The interface is designed as four text fields with buttons to search using ANY or ALL of the entered terms, these buttons make use of twitter's SEARCH API and either OR or AND the entered terms together respectively. This design was chosen as being the most effective way to hide the raw AND/OR logic from the user whilst still keeping the behaviour clear.

It was decided that retrieved tweets would be displayed in a format as close to tweets on twitter as possible. Each tweet therefore has a link to the author's page, the time and date it was made and a link to the original author if it is a retweet.

We had to make sure that the statistics for the tweets were calculated in an efficient manner. When data is retrieved from the server the tweets are parsed through once. During this parse all logic required to both display the tweets and extract relevant information for statistics is carried out. After this initial parse the top 20 words for example can be found by sorting the array of words by their frequency and taking the top 20. The advantage of this solution is that all of the data required is collected in one parse through of the tweets - saving unnecessary looping over large amounts of data.

We have used Twitter's search API to gather information about tweets as it provides a large collection of data that, when used with the proper queries, is very relevant. We decided not to use the Twitter's streaming API as it would require the server to keep connection to API if we want the most relevant tweets.

1.1.3 Requirements

To the best of our knowledge, all requirements for this section have been met.

1.1.4 Limitations

All work to calculate statistics is done in one parse of the data, this means that there is a small delay before any information is displayed. It may have been faster to display just the tweets as soon as they are retrieved and calculate the statistics separately.

1.2 Storing Information

1.2.1 Issues

The task for this section was to design and create a database capable of holding relevant information collected from queries. This database was then to be used as a source of data and as a cache to reduce the number of queries sent to the twitter API. The main challenges associated with this section were designing the database schema and implementing caching.

1.2.2 Design Choices

The database is designed around twitter users as they provide the relevancy to the tweets that we collect. All the tweets are associated with the user that wrote it and then the rest of the data is associated with that tweet. This design would allow us to find specific areas of interest that a particular person is interested in.

The schema is broken down into elements of a tweet. This allows us to reduce the total amount of data needed to be stored as repeated data such as media, hashtags and users are only stored once as their occurrences are stored in a linking table. This further extends our ability to pull specific collections of data efficiently, for example, the total use of a known hashtag.

Also in the database we have stored the teams with the largest twitter following and all the teams currently in the premiership league as well as the players in those teams. We feel these are the teams and players most likely to be searched. This assists in validation as it allows users to search for specific teams and players.

1.2.3 Requirements

The requirement of storage information has been met however we feel that we will improve on the quality of the implementation.

1.2.4 Limitations

Teams and players that have been added to the database for validation must be added manually and have to be researched so that the information is correct and up to date.

Due to the data being spread between multiple tables for a single tweet, it can slow down the process of collecting the data to provide to users. This also means that the server will be kept busy for longer potentially stopping over users from using it while it is processing another users query.

At present, not all information about a tweet is stored. This reduces the amount of data that can be analysed or displayed when tweets are collected from the database instead of the Twitter search API.

1.3 Web Interface

1.3.1 Issues

The task for this section was to develop an interface in HTML and Javascript that would be served by a node.js server and allow the user to perform the required queries. The main challenges associated with this task were validation of user input and the overall design.

1.3.2 Design Choices

A simple design is used to keep the solution clean, Google's Oswald font is used throughout [5]. The interface is designed as four text fields with buttons to search using ANY or ALL of the entered terms.

Different text fields have different inputs. For example, users may only enter one team but several players, hashtags or keywords separated by commas. This gives the user the freedom to design a complex set of queries.

Validations are handled in one of two ways depending on the field. The database contains a list of team names along with their twitter handles. The team field must contain one of the teams that is in this database. A JQuery autocomplete plugin is used to suggest teams as the user types [6]. This way we can ensure that the team exists on twitter and the user can carry out a search without needing to know the specific twitter handle however it does introduce certain limitations discussed in section 1.3.4.

Players have not been validated at this stage.

Hashtags are validated to ensure that they don't start with a number, consist of only a number, contain whitespace or special characters in accordance with twitter's rules for valid hashtags. At this stage keywords are not validated, we felt it would be useful to give the user the chance to search for numbers or phrases.

The interface is served by a node.js server which carries out all interactions with the social web.

1.3.3 Requirements

To the best of our knowledge, most of the requirements for this section have been met. We are missing player validation currently and intend to include this in the final system in a way which will work similarly to the team validation.

1.3.4 Limitations

One of the main limitations of this solution is that the team validations are limited by the information that is in the database. Therefore to be used on a larger scale the system would require a thorough database input as well as long term maintenance. Furthermore, if the user knows the specific handle of a team that is not in the database they will not be able to use it.

1.4 Quality of the Solution

1.4.1 Issues

The task for this section was to produce a solution of the highest quality possible. The main challenges associated with this task were to translate information entered by the user into a query that would collect relevant tweets and storing large amounts of complex data in an efficient way.

1.4.2 Design Choices

We have taken some steps to improve the overall efficiency of the system. Firstly we use javascript to check on the client side that the information entered into the form is valid, and in the case of the team we check against our database that the team in question has a handle on twitter. Calls are made to the server as Ajax requests which means that the client can remain active while the server fetches the results. Information is then sent to the client as a JSON object [7].

By creating a simple user interface (4 fields and 2 buttons) the behaviour of the system has been made more transparent to the user. Furthermore, the way that results are presented to the user adds to the overall quality of the solution. We have displayed retrieved tweets in a manner as close as possible to twitter so that they will be easy to read and understand and used simple ideas such as bronze, silver and gold in our statistics pane to convey information more clearly. The google map, implemented as discussed in lectures [8], is fully functional but we have found during testing that very few users geolocate their tweets.

We have broken the complex data down to its basic information components, this has allowed us to ensure that there is minimal data repetition in the database.

1.4.3 Requirements

We believe that we will work on improvements to the quality of the solution before the final handin. We would like to incorporate the statistics panes in a different layout so that the user doesn't have to scroll down the page to see the tweets. We would also like to adapt the database design.

1.4.4 Limitations

When a user presses a button on the form there is a slight delay before results are displayed. It would have been beneficial for us to display a ‘loading’ message or similar so that the behaviour of the system was clear at all times.

The google map is implemented but hardly used. We are considering ways we could use other types of social media to collect more geo located data to make use of the map.

1.5 Additional Features

At this stage no additional features have been implemented.

Conclusions

We found this assignment very interesting as we had never done anything similar before, nor had we worked with node.js. We had to think about the best way to present the information to the user, we decided to implement a statistics pane with bronze, silver and gold entries as well as a twitter like feed.

Due to the nature of node.js the server runs efficiently even when dealing with large amounts of data.

Division of Work

We both agree that there has been a 50/50 division of work. As the assignment had a client and a server side this seemed to be the most natural way to divide the work.

James worked on the server side, writing the code for the node.js server and querying twitter. He also produced the database.

Nicola worked on the client side, writing the code to display tweets, calculate their statistics and display geolocated tweets on a google map.

Documentation was produced by us both.

Extra Information

The server has been submitted with a config file that contains the connection information for a MySQL database and our Twitter API. This will be needed to change to the correct information for testing purposes.

Bibliography

[1] As detailed in the handout for the assignment on page 3

[2] https://dev.twitter.com/rest/public/search

[3] Lecture Week 2, https://vle.shef.ac.uk/bbcswebdav/pid-2205917-dt-content-rid-4890265_1/courses/COM3504.A.154617/week%202%20accessing%20the%20Web%284%29.pdf

[4] Lecture Week 3, https://vle.shef.ac.uk/bbcswebdav/pid-2212472-dt-content-rid-4890263_1/courses/COM3504.A.154617/Week%203%20Web%202.0%283%29.pdf

[5] https://www.google.com/fonts/specimen/Oswald

[6] http://jqueryui.com/autocomplete/

[7] Lecture Week 5, https://vle.shef.ac.uk/bbcswebdav/pid-2222544-dt-content-rid-4916571_1/courses/COM3504.A.154617/Week%205%20socket.io%20WebRTC.pdf

[8] Lecture Week 4, https://vle.shef.ac.uk/bbcswebdav/pid-2217145-dt-content-rid-4903590_1/courses/COM3504.A.154617/Week%204%20Understanding%20Web%202.0%282%29.pdf