Cluster and Cloud Computing Assignment 2
Twitter analytics of Birmingham,UK

Team 17
Abdon Carrera - 645998
Richard Gale - 319740
Rocio Mera - 647650
Ivan Malo - 661332
Siyuan Zhang - 668506

Abstract

This report presents the works that have been done by Team 17 for capturing the essence of life in Birmingham, United Kingdom. First, the design and architecture of the developed system are explained in detail, followed by a section explicitly illustrating the rationale of using various components in the system. To help readers understanding how our works helped to reveal the life of birmingham residents, we presents a number of elaborate scenarios that captured people's daily sentiment and hottest topics they are interested in. Other than these interesting facts of birmingham, the report also demonstrates how a big data analysis task can be done via cloud computing.

Introduction

Modern society has overwhelmed by big data. Mining the information and knowledge behind these data delivers great business and research values to companies and organizations. Twitter as a popular web application that has more than 302 million active users monthly[1] generates massive amount of data daily. These data can be used as evidences and research materials in human emotional and behavioural researches. Nevertheless, handling large amount of data is generally slow in single machine, thus cloud computing is a better option. This cloud based project is motivated by these and aims to reveal and capture the essence of life in Birmingham, United Kingdom by analyzing large volume of tweets using the computing and storing resources of a public cloud. In order to do this, the team first used program written in Python to harvest tweets from both Twitter Streaming API and Twitter Search API, then store the collected data into CouchDB. Before the data is permanently saved in CouchDB, they were sent to MeaningCloud where the data was analyzed and sentiment analysis of given message were attached. Those sentiment attributes formed the foundation of the analysis in this project. To better understand the essence of life in Birmingham, the team carefully prepared a number of scenarios that helped to reveal the sentiment and interests of Birmingham residents. MapReduce[2] functions were used in the project to transform the tweet data into statistics and knowledge. By doing this, the team acquired a clear view of various aspects of the life in Birmingham. Through this project,

the team was able to gain greater understanding of working with cloud computing platform, Schemaless databases, automation tools, RESTful APIs, and the cloud computing paradigm as a whole. Certain aspects of life in Birmingham was discovered, including the political sentiments of Twitter users corresponding to the results of the General Election, as well as the relationship between Birmingham people and the Birmingham accent.

This project is the work of five team members. With overlaps across multiple roles, the main responsibilities of each team member were:

1. Abdon Carrera - Implementation of Tweet harvesting using Twitter search API, Web RESTful service, Front end web site.
2. Richard Gale - Integration of Meaningcloud with the tweet harvester, implementation of the MapReduce functions for each of the scenarios covered, integration of the MapReduce functions with the Python program to create views and their subsequent analyses
3. Rocio Mera - Cloud setup, CouchDB setup, Ansible automation(software environment), Front end web site, creation of services/scripts from all the code available (Java/Python programs)
4. Ivan Malo - Research of the Twitter API documentation and city culture, implementation of the harvesting application and design of scenarios.
5. Siyuan Zhang - Implementation of MapReduce functions for scenarios, implementation of python program for creating views and subsequent analysis.

System Functionalities

Our goal was to design MapReduce functions to analyse the various scenarios that exist in the city of Birmingham. In order to achieve this goal, we created a system that realised these main functionalities:

-Automated setup of couchDB database in multiple nodes in the Nectar cloud. All setup and dependencies are installed using Ansible.

-Scheduled collection of tweets using a harvester program utilising the Twitter API, to selectively harvest tweets from the city of Birmingham. A separate service is used to run these jobs, as well as to attach Meaningcloud attributes to the tweets.

-Python program to integrate the created MapReduce functions to the views inside couchDB.

System Design and Architecture

In this section we present in detail the design and architecture of the harvesting/sentiment analysis system implemented in the NeCTAR Research Cloud. In order to provide a better understanding to the reader and explain in depth the details of the system design and architecture, this section look at different parts of system (Figure 1) including the components of the cloud, nodes and databases; the architecture of the tweet harvesting process; the approach of the replication of the databases; the module of sentiment analysis; the architecture of the web services and web server and front end

interface. Moreover, in the discussions below we give some insights of how we adopted them and why we select them as our final architecture and design .

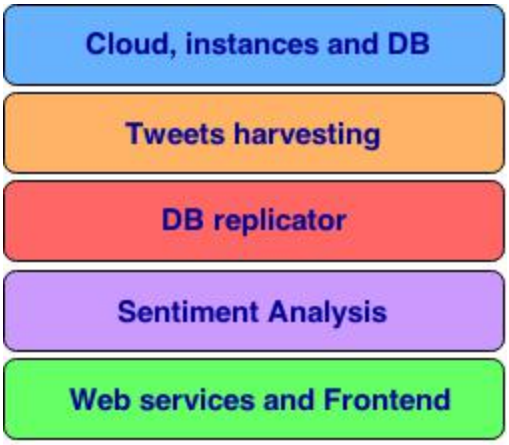


Figure 1: Cloud-based solution components

Cloud, Instances and database

For this project we used NeCTAR Research Cloud and deployed a cloud-based solution using five instances. We decided to use four instances (Node 1-4) to manage the harvesting and storage of tweets is used to provide web services and manage the web server.

Three (Node 2, Node 3 and Node 4) of the four instances destinated to harvest tweets have the following characteristics: VCPU = 1, RAM = 4GB, Storage = 40GB. Those instances have small characteristic because they are only gathering tweets and storing them in a database. Thus, we do not think that they require more RAM or more CPU. On the other hand, Node 1 (considered as a database server) and the web server node have better characteristics because they are processing information, analyzing content and they will be used to reply all the request from different clients. Their characteristics are: VCPU = 2, RAM = 6GB, Storage = 60GB.

The operation system of all the instances involved in the cloud is CentOS 7 because most of our team members already has experience working under the mentioned operation system. Moreover, CentOS is well used as a server platform and also for systems that offer services.

The database used to store and manage the manipulation of the tweets is CouchDB, since it offers an efficient way to handle big data and also due to the advantage of manipulate documents in JSON format which helps us to store the data directly from the twitter API without creating a scheme (More details are presented in the couchdb section).

The last component is the web server. This node contains an Apache Tomcat web server and uses Java JAX-RS (Jersey) to provide services and respond resources. Figure X. shows an overview of the main components in our cloud-based solution.

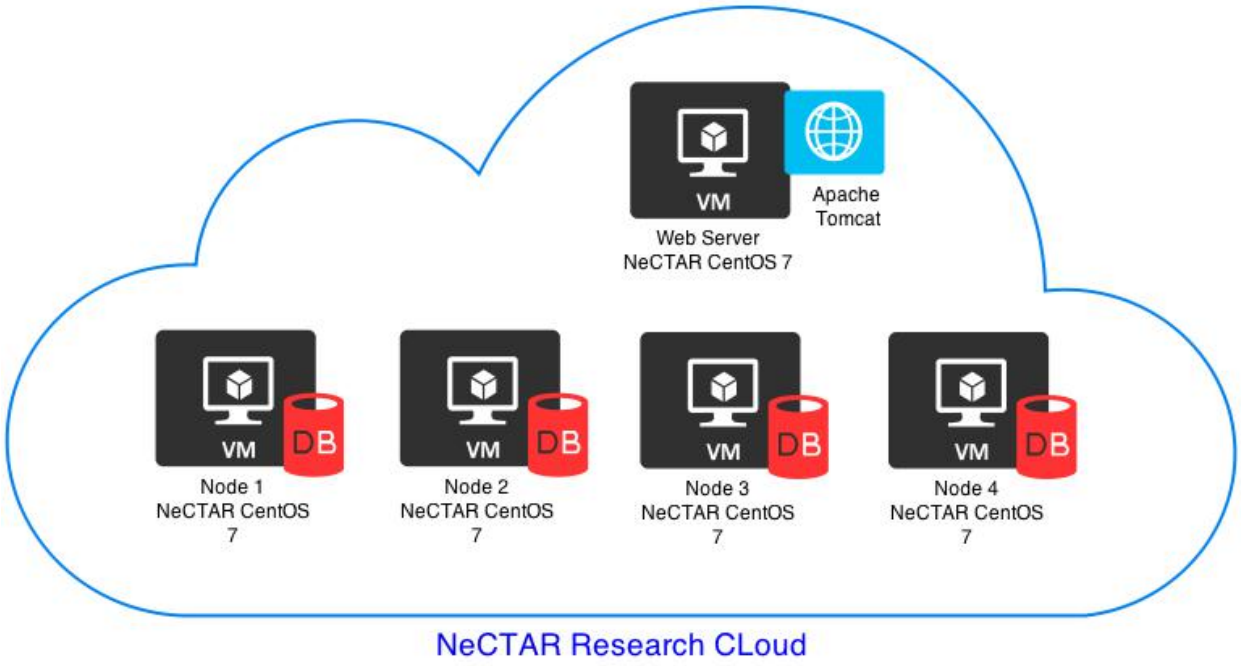


Figure 2: Node setup under the NeCTAR Research Cloud for harvesting tweets

Tweet harvesting design and architecture

We selected four nodes to be responsible for the collections of tweets and store them on a CouchDB database. Part of the design is to use four different nodes so that each node represents a section on Birmingham. We came up with the idea of dividing Birmingham's city into four sections, so each of the four nodes can collect data from a different region of Birmingham. The goal of this approach is to facilitate the manipulation of data, reduce the load on different databases and to analyze data faster in different regions of Birmingham.

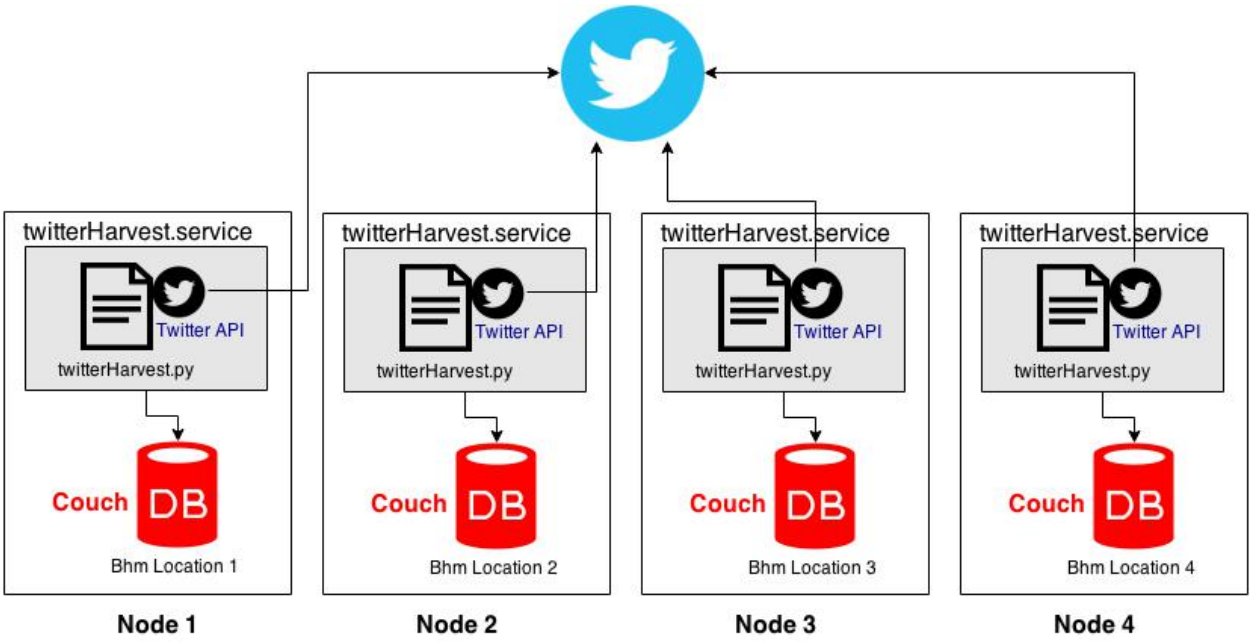


Figure 3: Details of the tweet harvesting architecture

Figure 3 illustrates the architecture's approach that we use for harvesting the tweets. The collection of tweets is done by using a service (twitterHarvest.service) on each node. The service runs a python program (twitterHarvest.py) that calls the twitter API. We decided the idea of making a service from the python code because when the machines are deployed using ansible, we want them to start the service so that the collection tweets is immediately.

We used several approaches to get the many tweets from Birmingham, some of the approaches include using streaming API to get tweets based on Birmingham's location using coordinates other approaches include using the search API to look at keywords, phrases, hashtags, users, etc. (more details in the Twitter Harvester Application section).

The information returned from the twitter API is in JSON format, therefore using the same service (twitterHarvest.service) and python program (twitterHarvest.py) we proceed to check the tweet and store them in the node's database. The format of the tweet stored in the database will be the same JSON provided by twitter with all the attributes of the tweet.

DB replicator

Part of our system's design is to obtain consistency in the information spread on different nodes in different databases. Therefore, as a method to obtain a strong consistency of the tweets, as well as to remove duplicate tweets, we decided to synchronize all the databases from the four nodes into a single point that will act as a Database server.

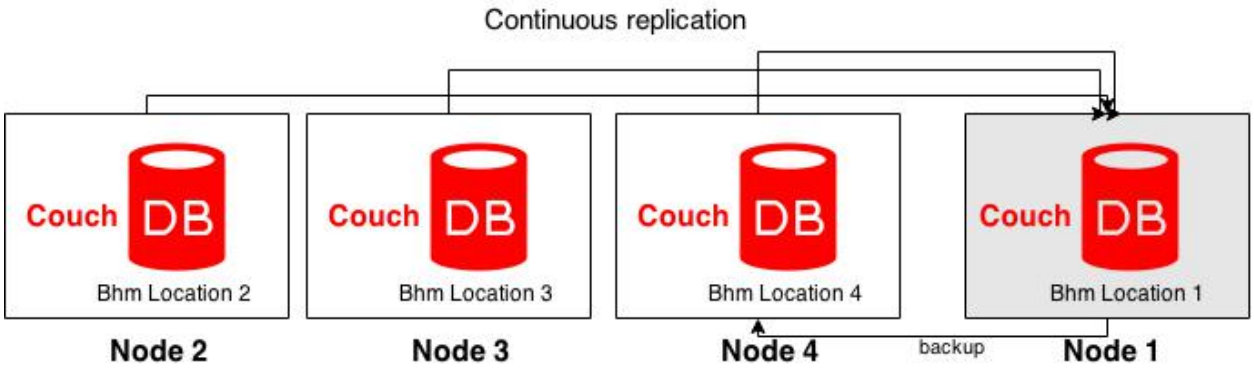


Figure 4: CouchDB replicator mechanism across different nodes

Figure 4 shows the replication of the location's databases into the Node 1 database. We use Node 1 as a master node, it helps for the communication between the web server and web service. One benefit of the replication from different databases into a single database is the removal of duplicate documents. In this scenario the reliable synchronization between multiples machines will help to avoid the redundant data storage. As we set the tweet ID of as the document ID of the JSON document stored in the database, if there are duplicated tweets with same tweet ID in different databases, the replication will remove the duplicated information resulting in a consistent database without redundant tweets.

For this reason in our design, we created a single side continuous replication for the different nodes to the master node. We also considered fault tolerance by having a backup of our tweets, consequently we make use of the replicator module to replicate our information in other database on a different machine.

Sentiment analysis

Our system's design also contains a sentiment module for analyzing tweets. The architecture of the module is represented in figure X.

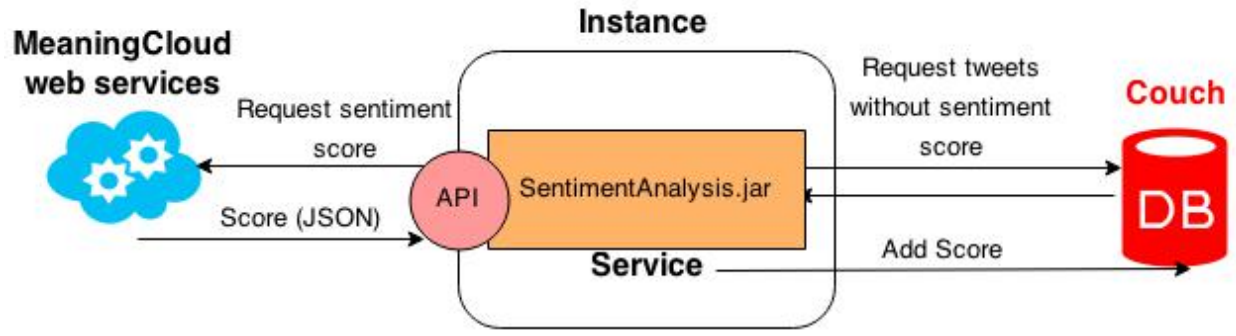


Figure 5: Meaningcloud addition, sentiment analysis architecture

This sentiment module is part of the overall architecture of the cloud-based solution. The benefits of integrating this module is that it will automatically and call the Meaningcloud API[3] for a sentiment

score of and a list of topics for each tweet on the database. This approach will allow adding data to the tweets on the fly while gathering them at the same time. With this approach, we can analyze different tweets from different machines in parallel resulting in less overhead of processing and analyzing the tweets.

The architecture of this module includes the use of an external web service provided by Meaning cloud (More information in the Meaningcloud section), the creation of a service that invokes an executable .jar program (SentimentAnalysis.jar) and the database containing the tweets that are going to be analyzed.

On couchDB, we have a view that identifies the tweets that already possess the sentiment score and other view that identifies tweets that have not been analyzed by using the meaningCloud API. The Sentiment Analysis service starts looking at the view with the tweet's document that do not contains a sentiment score. For each tweet, the java program extracts the "text" attribute contained in their JSON and calls the services from the MeaningCloud API. Once the MeaningCloud web service has analyzed the text of the tweet, it will reply a JSON containing the analysis and a sentiment score. Then, the Sentiment Analysis service will update the database adding the corresponding sentiment score at each tweet document.

Web service and frontend

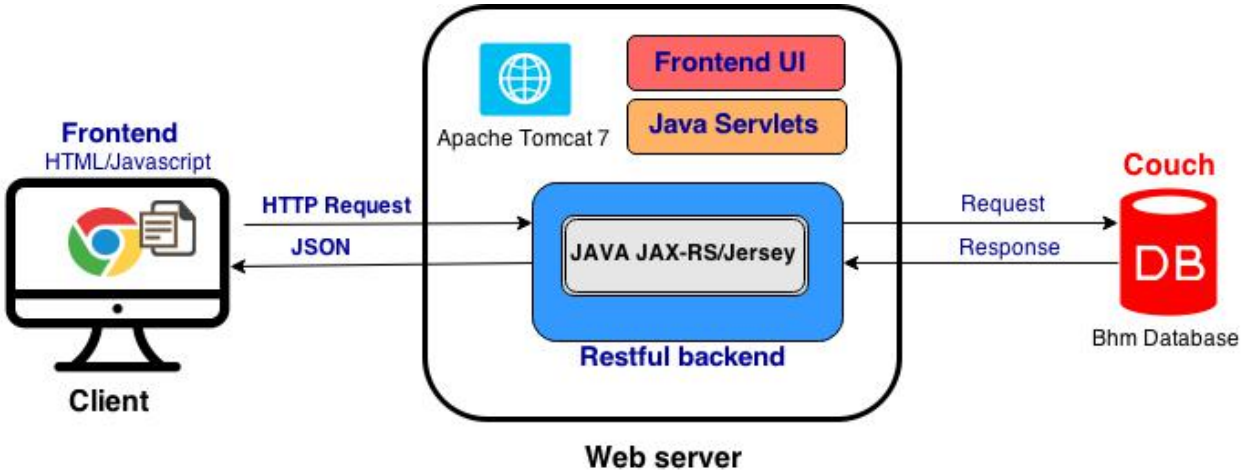


Figure 6: Web server architecture

The architecture of the web server and its main components include: A frontend website (client side), HTTP requests, JSON as reply, Java Restful web service[4] (JAX-RS), Java servlets, Apache Tomcat 7 and the CouchDB database (Figure X.X).

We have adopted a RESTful web service architecture which consists on a Java web service using JAX-RS(Jersey). We chose Java Jersey for this approach because it makes easier the development of the web service and offers simple ways to work along with Java Servlets.

The frontend application includes HTML and Javascript to display all the information returned from the web service. For example, lets see the following scenario:

- The client using the frontend website asks resources by using HTTP Request to the REST web service.
- The web service receives the request and using Java servlets forwards the request and ask to the couchDB database.
- The couchDB returns the views/documents with the information required from the java servlet.
- Then, the web service proceeds to reply the client via JSON.
- Finally the Frontend websites displays the information of the JSON using javascript or HTML.

Scalability of the application and Infrastructure

With Nectar cloud we can deploy new instances, add or remove resources as needed. We also have some modules like web servers, databases, sentiment analysis and tweet harvesting available from our ansible script that can be easily adapted to the new instances and to increase our infrastructure (scale horizontally).

However, due to our original design of the cloud-based solution shows a centralized database architecture, it might not meet the desirable scalability. We are aware that we might have a bottleneck on a high demand of users accessing the application. Moreover we have a fixed web server asking petitions to a single node (data server).

Ideally, it would be better to have a load balance manager. For example, if we deploy two new server to the web application into our cloud solution (resulting in three servers), the load balance manager would help to distribute the workload between the different servers available. The same example of load balance can be applied to the CouchDB databases.

System Components

In this section, components that comprise the system will be explored, including MeaningCloud, CouchDB, Twitter Harvester and NecTAR Research Cloud. Benefits of using these components will be explained in details as well as the challenges brought by their defects we encountered during development.

Twitter Harvester Application

Twitter provides developers a set of methods for access to its objects and build applications in creative ways. A twitter harvester application was built to storage a large volume of tweets on a database based on a specific criteria (by location and key terms), and performing sentiment analysis on the data. The application was based on three main components from Twitter's API[5]: OAuth authentication, REST API and Streaming. Two differents scripts were deployed to request calls to the Streaming and REST API. A mechanism to handle exceptions was coded to mitigate possible errors such connection/authentication errors or excess in the numbers of calls to twitter's API.

Twitter provides an alternative to process real time tweets. The Streaming API permits to push tweet data in real time from Twitter's global stream using a persistent HTTP connection open. It provides three different streaming endpoints for different implementations scenarios. The public streams endpoint is suitable for gather data from specific locations or topics, user streams endpoint allows to track a single user's view of twitter and site stream endpoint allows the application to monitor twitter streams for a large number of users. Unlike REST API, the streaming API has low overhead involved during the communication.

The twitter harvesting applications was coded in python version 3. The main library used for this purpose was Tweepy[6]. Tweepy is a simple-to-use open source library for accessing Twitter APIs. The script tweetharvester.py was used in 4 nodes to collects real time tweets from Twitter stream API. The script was designed to request the parameters location and track provided by stream API.

Location

The filter method provided by Tweepy was implemented to filter the tweets from the global stream using the locations request. Twitter requires a list of coordinates pair (longitude and latitude) in order to deliver only the tweets that are generated within this area. The bounding box contains four pairs of decimal coordinates starting from the southwest location. The exact coordinates from Birmingham (-2.0336485,52.381053,-1.7288577,52.6087058) were obtained from the website BoundingBox.com[7].

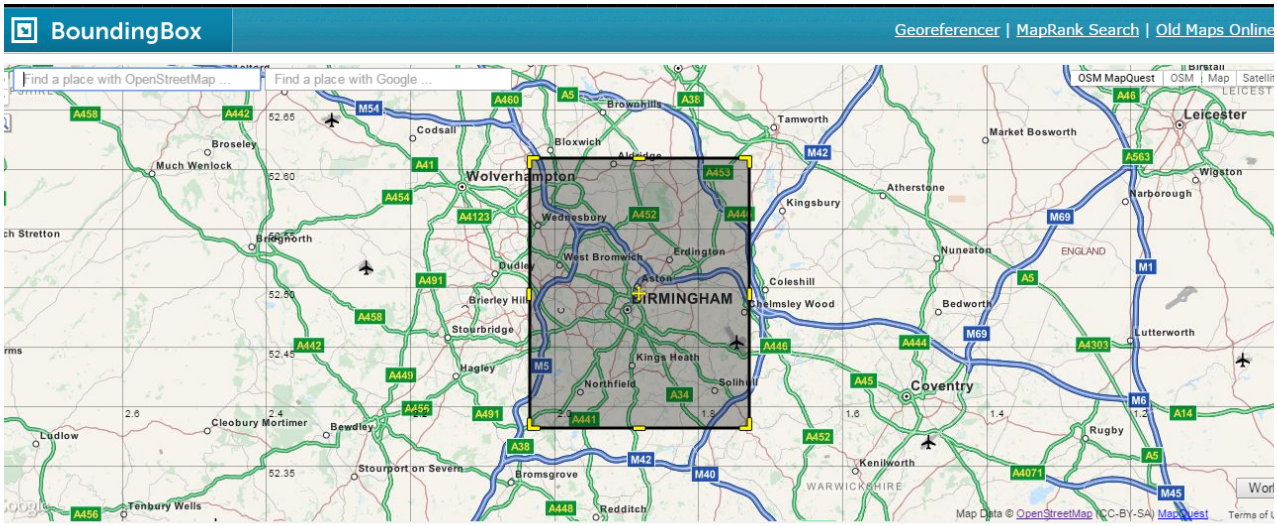


Figure 6: Coordinates of Birmingham (-2.0336485,52.381053,-1.7288577,52.6087058)

The bounding box of the whole city was divided in four quadrants for load balancing the data received from the stream on the four harvester nodes (table 1). Twitter evaluates whether a tweet was posted from a specific location by: checking if the coordinates field from the status object is not null and comparing if the point falls within the bounding box. In case that the coordinates field is empty but the place field contains a region (e.g Birmingham), Twitter will determine if the value of the place field overlaps with the bounding box. Finally, If none of these cases are true, the tweet location was generated outside the limits of the bounding box therefore is discarded.

The first tests of tweets recollection deployed on four nodes showed that most of the tweets were repeated in each of them (duplicates). Twitter does not guarantee an accurate match of location and recommends to implement an additional mechanism of filtering location. Another issue observed during the initial test was to determine whether a user with the place field populated with the Birmingham location was updating status from a different city or overseas. This could also be solved by ignoring tweets with place attribute and empty location.

Node	Quadrant	Coordinates
1	1	-2.033649,52.381053,-1.887473,52.497652
2	2	-1.887473,52.380634,-1.728858,52.497652
3	3	-2.033649,52.494074,-1.887473,52.608706
4	4	-1.887393,52.490729,-1.728858,52.608706

Figure 7: Coordinates in 4 segments for 4 nodes

Keywords

Another approach was used to further collect tweets with unidentified coordinates. The track filtering parameter allows to receive tweets from the stream filtering by phrases or keywords. A match occur only if the keyword or all words from a phrase appears in the text field of the tweet. A selection of characteristic phrases and hashtags from Birmingham were selected to increase the number of tweets in the database. The key terms were divided into categories:

Category	Keywords/Phrases
City	'Birmingham', 'Birmingham city', 'Birmingham City Council', 'Brummies', 'Brum', 'Brummie', 'Brummagem', 'Bromwicham', 'Birmingham's', 'bham'
Politicians	'Shafique Shah', 'Sir Albert Bore', 'Albert Bore', 'Richard Burden', 'Liam Byrne', 'Jack Dromey', 'Roger Godsiff', 'John Hemming','Khalid Mahmood', 'Shabana Mahmood', 'Steve McCabe', 'Andrew Mitchell', 'Gisela Stuart'
Places	Barber Institute of Fine Arts', 'Lickey Hills', 'Clent Hills', 'Walton Hill', 'River Tame', 'Longbridge', 'Tamworth', 'Staffordshire', 'Lichfield', 'Warwickshire', 'Redditch', 'Sutton Park', 'Project Kingfisher', 'Woodgate Valley', 'CBSO Centre','Winterbourne Botanic', 'Digbeth Institute
Nightlife/Festivals	Brindleyplace', 'Newhall Street
Universities	Aston University', 'University of Birmingham', 'Birmingham City University', 'University College Birmingham', 'Newman University
Sports Teams	Aston Villa, Birmingham City F.C. , Moseley R.F.C., Moseley, BCFC, BCFC_Community, AVLEVE, astonvilla, avfc, AVFCOfficial
Elections 2015	'votelabour', 'ivoted', 'electionday', 'generalelection', 'general', 'election', 'vote','labour','2015', 'election2015','ed_miliband','uk', 'uklabour', 'miliband', 'david_cameron', 'cameron','brumvote', 'BrumVotes15','GE2015','GE15', 'ukip', 'messi'

The challenge with this approach was to not collect tweets from other cities from UK or others countries. The test results showed that a considerable number of tweets were originated from other cities especially in the city category(several tweets from Birmingham, Alabama were delivered to the application). A further post filtering was deployed to reduce the tweets from non Birmingham, UK origin.

A second script `tweets_from_users.py` was implemented to download tweets from Twitter REST APIs. The user-application authentication mode was selected for obtain data from the friends and lists on behalf of a particular user. It was used to recollect 200 last status from a list of 'well-known' Birmingham users. The list consisted 96 users and the selection criteria was based on:

Local news media: newspapers, radio station
Local Business:restaurants,cafes
Local government entities: universities, theatres, police
Local events promotion accounts (e.g. what to do in Birmingham)
Users who are related to any activity in the city (athletes, players, politicians, artist, professors)
Users whose profile bio express affinity with the city
Users who work in a local business

Additionally, the script was also used to download small samples of tweet data for test purpose in a local computer with the map-reduce functions offered by CouchDB

Error Handling

Both the Stream and REST APIs need methods to handle errors. Connection to the public stream are not limited. Network congestion and high latency during the stream connection is most the common issue that can impact the performance of the application. Twitter documentation alerts that multiples attempts of connection to the Stream could block the IP of the application server. To tackle this issue a time sleep method was deployed in case a connection error was received. The application will wait 5 second before trying another reconnection.

REST API requests are rate limited in a window of a 15 minutes. The solution for this problem was simple. Tweepy library includes methods to poll Twitter API and determine how many time it has to wait until request a new round of request (generally 15). It also can handles error of authentication (401).

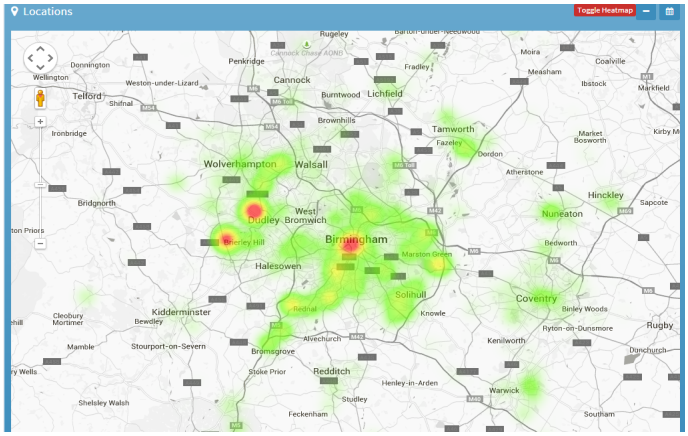


Figure 8: Heat Map of tweet concentration in Birmingham, as seen in the frontend application

MeaningCloud

In the project, the core parts of sentiment analysis and topic extraction are primarily dependent on MeaningCloud which is a cloud-based web service providing precise, accurate and multilingual sentiment analysis on given text[8]. Its powerful sentiment analysis API enables us to analyze the sentiment of each tweet in both global and attribute levels as well as extracting facts and opinions from the given text and detecting the polarity and irony of the message. In detail, it identifies the concepts and entities in the given text and assigns sentiment score to each of them based on natural language processing techniques. Main difference between entity and concept is that an entity is a specific person or place or event while concept is high-level abstract noun a.k.a. hypernym. When searching for certains topics, these two are both highly useful as entities helps to search for specific topics (e.g. project manager) whilst concepts can be used in searching of general topics (e.g. job).

One major limitation of using MeaningCloud for sentiment analysis is the access limit of API calls. Since MeaningCloud gives only 40,000 API per month calls to each account, the team had to register multiple MeaningCloud accounts for analyzing 10GB tweet data within a short time period of 3 weeks. And due to the extra overhead incurred through the API call, the rate of adding the Meaningcloud attributes was slower than the rate of the harvesting of Tweets. This resulted in not having all tweets not having the sentiment analysis data (approximately 50%).

Another limitation of MeaningCloud is that its sentiment score only reflects the overall sentiment of the message rather than the sentiment towards each topic that has been mentioned in the message. For instance, the tweet of "Although I dislike David Cameron, hearing the victory of conservative party still makes me excited." could have a positive sentiment score as overall the writer seems to be happy, despite a negative attitude towards David Cameron. Such cases reduces the accuracy of the overall sentiment analysis as they introduce anomalies to the dataset. As mentioned previously, MeaningCloud returns a list of entities and concepts but actually none of them could help when counting the number of mentions of a particular topic. For example, when counting the number of mentions of each Aston Villa football players, neither the entity list nor the concept list would help because MeaningCloud is not concept aware enough to recognize football players. So in this case, we hard-coded a list of avfc players that we aim to search for and search each of them on all tweets with filtering out those that do not contain "#avfc". Such a method may not be as helpful in finding out all tweets that mentions a avfc player, but at least it guarantees all returned tweets are about avfc, not anything else.

CouchDB

CouchDB is a NoSQL database that stores data with JSON documents. It natively provides RESTful HTTP APIs to access data and supports querying with MapReduce using Javascript.

Our project benefits from the following key features of couchDB:

1. HTTP-based RESTful APIs
   When CouchDB is installed, it can be easily accessed via port 5984. We are able to access the database and perform CRUD operations on data using simple HTTP requests. In the project, we use a python package called couchdb-python which provides useful modules for remote access of couchDB.

2. Futon
   CouchDB provides its users a web-based administration console called Futon. It gives us a clear visualization of data and design views. As it allows us to create temporary views in the database, which creates a design view that querying data using MapReduce functions but without saving it to the database, we can quickly test and debug our MapReduce functions on the real database without wasting space for storing it. It also presents JSON documents in a more readable format so that the time of converting JSON string into human readable structure can be saved.

3. Document-oriented storage
   CouchDB uses schema-free JSON documents to store data, which means documents stored in CouchDB are like the real-world document, self-contained, having no unified data model/structure. In contrast to relational database, in CouchDB, json document is the basic storage unit rather than entities. When storing a document, CouchDB could store it directly rather than dividing it into smaller pieces with logical connection within them and inserting each of them into a separate table. This design makes it more scalable and flexible.

4. MapReduce
   MapReduce in CouchDB can be use to query data in parallel. Each document in the database will be filtered and sorted by Map() function and summarized by Reduce() function. This technique is quite powerful and efficient in processing large amount of data such as twitter data in our project. For more than 10 GB twitter data, a traditional database could take considerable time to process while in CouchDB, the data can be processed in parallel using Map() function. It could take relatively long time to build B-tree for a view but it will cost much less time for querying once the tree structure was built. Another benefit of having the B-tree structure is that it is scalable, meaning when new data were added to the database, CouchDB will only perform calculation on those new data rather than rebuilding the entire tree unless the MapReduce function changed. In this project, we use Map() function to filter data and count number of occurrence of targeted topic or sum up sentiment score with Reduce() function.

5. Replication
   A great feature of CouchDB is its support of replication function. In our project, we utilize it to eliminate duplicates by moving all tweets to a selected node after tweets from different areas are collected by other nodes. The replicator will tackle all duplicated tweets when copying from one database to another since tweets with identical IDs will be automatically replaced.

6. No Locking
   Another advantage of using CouchDB is that it has no locking schemes. In contrast to relational database, CouchDB does not block users reading data when it is being updated by other users. Instead of any locking mechanisms, CouchDB uses Multi-Version Concurrency Control (MVCC) to organize concurrent access to database from multiple users.

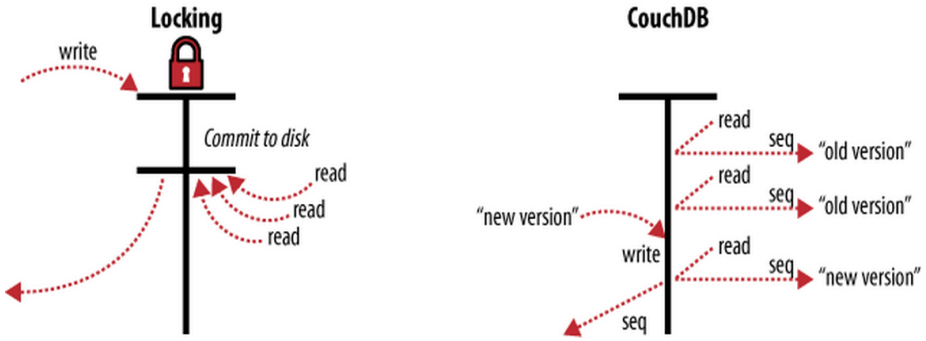


Figure 9: Locking mechanism in CouchDB[9]

As can be seen from above figure, CouchDB will generate different version of data when updates occur. This feature helps to effectively utilize the processing power as clients won't have to wait while the system is processing some other requests. With this feature, the system is able to perform analysis on existing data while the harvester is still adding new data to the database.

Apart from the great number of benefits that couchDB brings, it also contained drawbacks which made the MapReduce analysis more difficult:

1. No built-in full text search
   Since CouchDB natively does not support full-text search, the data processing becomes less efficient. Querying in CouchDB can only be done by MapReduce function that is applied to all documents in the database. It introduces redundant steps when searching for a particular document. Although integration with some tools (e.g. couchdb-lucene) may help, it added extra workload to the team. In the project, we splitted tweet text into words and try to find a match between a selected keyword and each of these words. As this iterative search is performed on each of tweet documents, the performance is poor when searching for a document containing particular word on big data.

2. One view per query
   Again, despite the huge convenience MapReduce has given us, it makes testing processes extremely difficult. As mentioned above, querying in CouchDB can only via design views which the system on the main node will take few hours to compile gigabyte-sized data. It means when testing on the main node, it will take few hours to get feedbacks as it will cost such long time for the system to process each view. To overcome this, we tested views on some smaller sized databases before uploading to the main node as less view processing time for smaller database. However, smaller sized databases lead to another problem. As there are less

tweets in small sized database, some queries may get empty result returned. For this reason, we have to create ad-hoc databases for each query. For example, when testing the MapReduce function that querying the number of mentions of avfc players, we have to use the harvester to collect tweets about avfc and create a new database to store all relevant tweets, since the database for general purpose does not always give satisfactory results. This is one of the main difficulties we encountered in data collection and data analysis phases as creating ad-hoc database for each query consumes considerable time.

3. No Sorting by values
   CouchDB by default does not have any value-based sorting mechanisms. When counting the number of occurrences of categorical data, it will just return an unordered list of key-value pairs. In order to find out the top ten most mentioned items, we needed to fetch the whole list of key-value pairs and feed them to a sorting program coded in Python. Generally, for data of size of 10 GB, it will takes few minutes to sort returned list. It is inefficient and introduces a single bottleneck to the system as it depends heavily on the data transfer rate and computation power of the machine running the sorting program. In the project, the results returned by each view are sorted/filtered locally and then uploaded to a separate database on the node. The web analytics application will read results from this database instead of views because reading results directly from views is impractical as it takes minutes for reading and sorting while the database storing sorted data will give instant result.

4. No chained MapReduce Views
   Unlike its extension, the CloudAnt, CouchDB does not get benefits from chained views as it natively does not support it. Chained Views allows user to build views based on the results returned by views, which can be used to examine data in more complex ways. For example, it can be used to sort data on value by having one view querying relevant documents and another sorting on returned values. For this limitation, our team can only perform simple queries on data and process them later in the Python program.

NecTAR Research Cloud

NecTAR is the acronym of National eResearch Collaboration Tools and Resources[10], which provides all Australian researchers with powerful information and technology infrastructure (ICT) that helps them to promote the research collaboration and research outcomes. In this project, the system we developed is hosted in the NecTAR Research Cloud which is an OpenStack based computational cloud. The Research Cloud provides its user up to 8 high-performing nodes that can be used to host research applications. As one of the biggest OpenStack based clouds, it benefits from the strengths of openstack in following aspects.

1. Dashboard
   It provides its user an interactive user interface to manage the computation, storage and networking resources, allowing users or administrators to have an overview of the current

resources usage and current VM instances. All operations can be easily done in the interface without editing complex configuration scripts.

2. Security
   It allows user to customize different security groups to access the created instances. It supports SSH communication with reliable RSA public-key scheme for system management and applications uploading and HTTP communication for web services. For authentication, it provides robust access control. The access to the instance can be controlled in user, role and project levels and it can be done via username/password based authentication (openstack) or token-based authentication (ec2).

3. Storage
   It gives its user freedom to customize the storage of each instance. When an instance is created, the cloud will automatically allocate an ephemeral storage (10GB) to it. User could also allocate persistent volumes to different nodes by need. In the project, we allocate 80GB to the main node and 40 GB to each of other nodes as the main nodes stores all tweets from birmingham while other nodes store a fraction of tweets.

4. APIs
   A range of graphic, command line, application interfaces are available in NecTAR Research Cloud. User can use use these APIs to better support the research applications.

Despite the immense benefits in Cloud computing provided by the Nectar Research cloud, some of the challenges/disadvantages applicable to the project were:

1. The lack of elasticity available for sudden load increases
   Due to the manual setup of the nodes, the Nectar infrastructure does not respond effectively to a sudden rise in the access to the system unlike EC2[11] provided by Amazon. This was not a direct issue encountered in the project, however it is problematic for other works with a combination of a high reliability requirement as well as high network demand.
2. Does not provide Microsoft Windows as an operating system for the nodes Although it did not affect this project in a significant way, other works requiring specialised Windows software will be affected by the licensing restrictions on Nectar.
3. Does not provide direct access into the Operating System inside the nodes
   The only interface available to communicate with each nodes is the command line terminal, meaning that one cannot access the OS GUI of the node directly. This may mean manual setup of the node is more cumbersome, requiring extra steps that would otherwise be made easier using a graphical interface.
4. Reliability issues/ nodes going down
   Reliability issues were encountered throughout the harvesting process, with the nodes going down multiple times due to system issues on Nectar. This affected the schedule of the harvesting process and the project as a whole.

Scenarios

General Mechanics of MapReduce Functions and Python

There were a multitude of scenarios covered using MapReduce functions on the harvested twitter data stored in couchDB. Some functions simply emitted the tweets which satisfied a certain condition, while for others, the default reduce functions of '_sum' and '_count' were used in conjunction with Python processing to aggregate properties and calculate the average sentiment for certain scenarios. The Meaningcloud attributes were used for sentiment analysis, namely the sentiment score of each tweet from -1 to +1, as well as the topic attributes which were the list of entities and concepts for that tweet extracted by the Meaningcloud service.

An example of a scenario covered by MapReduce in couchDB, along with post-processing with Python is the following:

```
function(doc) {
      topics = ['accent']
      user_name = doc.tweet_data.user.screen_name
      if (is_mentioned(topics, doc)){
            if (doc.meaningcloud.score){
                  score = doc.meaningcloud.score
                  emit('Accent tweet', [1, parseFloat(score)]);
            }
      }
}
```

Figure 10: part of the Map function for the view `'topic_accent'`

This function goes through the tweet to find if 'accent' is mentioned anywhere in the tweet. If it is mentioned, it emits the key of 'Accent tweet' and a list as its value, with a 1 and the sentiment score of that tweet. When the reduce function '_sum' is used in couchDB, all of the results with the same key is aggregated and summed, meaning that the final product is a single 'Accent tweet', with [total number of tweets, total sentiment score of all tweets] as value. A Rereduce-like operation is done in python, as other views often require the sorting on values (not readily available on couchDB), and in this case list[1] is divided by list[0], to get the mean sentiment score. Thus, through these two types of processing, an average sentiment score of -0.093 was obtained for tweets that has 'accent' as the topic.

All of the MapReduce functions used to create views can be accessed in the folder /map_reduce_functions, inside the final delivery of the system. The views correspond to the Map

functions used to create them, which can be accessed in the couchDB database. Here are some examples scenarios which were covered, and their analyses:

Scenarios and Analysis

The Most Tweeted Hashtags

In analysing the life of Birmingham through the scope of tweets, one of the main focus was on what people talked about the most. This was implemented for the view 'hash_tag_topics', it was discovered that the following were the top 20 hashtags mentioned in the harvested tweets:

Hashtag	**Mentions**	**Hashtag**	**Mentions**
#ge2015	185558	#votelabour	5532
#birmingham	79523	#staffordshire	5232
#votesnp	32065	#generalelection	4953
#onstagewiththevamps	31478	#bhamplay	4750
#jobs	30155	#london	4659
#ge15	27832	#libdems	4261
#job	17876	#warwickshire	3995
#snp	12438	#labourmustwin	3944
#labour	10661	#voteukip	3873
#ukip	7227	#bcfc	3849

Figure 11: The top mentioned hashtags with the number of mentions, across approximately 1 million tweets

As it can be seen, since the harvesting period was from late April to mid May of 2015, the main topic of focus was the UK general election which took place on 7th May 2015. Out of the 20 hashtags over half of them were election related, those for support of different political parties across the political spectrum. After observing these hash tags, the average sentiment of the tweets were extracted using the '_sum' reduce function in conjunction with python processing to generate a new Map function ('hash_tag_sentiment'). It was observed that '#onstagewiththevamps' was the most mentioned hashtag that was nonpolitical, with an average sentiment score of -0.472. As an outlier, the mention of the music band 'The Vamps' was examined further, to uncover its place in the life of Birmingham.

The Vamps

Namely three map functions were implemented to gain knowledge about the vamps, 'vamps_most_positive' 'vamps_most_negative' and 'vamps_sentiment'. One observation was the noticeably low average sentiment score for the hashtag, which was consolidated with the discovery that any tweet which mentioned The Vamps had an average sentiment score of -0.54881, observed in the 'vamps_sentiment' view. This outcome was not consistent with the immense popularity of the band across the UK and the world.

'vamps_most_negative' was written with the motivation of finding the tweets which criticised the band, with the assumption that there were highly vocal people amongst the tweeting population of Birmingham that posted negative remarks of The Vamps across Twitter. However, it was found that firstly, The Vamps held a concert on May 8th in Birmingham, and an overwhelming majority of the tweets with a negative sentiment of <= -0.8 were from repeated tweets of highly excited fans who were posting where they were in the stadium. Thus, it was an error in the classifier which judged these spam-like tweets to be of negative sentiment, which clearly was not. Thus in the course of this search, the unexpected finding was the loud and frequent voice of die hard Vamps fans in Birmingham posting their seat locations at the concert venue.

With 'vamps_most_positive', all tweets with a sentiment score of > 0.7 was collected, with the expected result of tweets which contained a strong praise of the band. Examples include:

["@thevampsband @thevampstristan can't wait to see you on friday at birmingham!!!□☺ #vampssaythanks", "0.80"]

["they've sold out birmingham im so proud□ @thevampsband http://t.co/9cbq8tafew", "0.80"]

["@thevampsband #vampssaythanks can't wait to see you in birmingham □❤❤❤❤❤❤❤❤□", "0.80"]

Figure 12: Example positive tweets about the Vamps Band concert in Birmingham

Thus through the MapReduce functions, the general scenery of the fan base of The Vamps in Birmingham was uncovered, as well as the voice of those who were the most positive towards them, with the low average explained by the sheer number of fans tweeting their location at the concert venue.

General elections

Following from the discovery that the UK General Elections on May 7th was the most popular agenda in terms of Hashtags in the 'Tweetsphere', further analysis was conducted to find out more about the Birmingham tweeters' political sentiments. The views 'election_mentions' and 'election_sentiment' both captured the number of mentions of major political parties, based on the tweets which had #ge2015 or #ge15, at the same time mentioning the names of key figures of each party. Further, the summation was

divided into 3 time periods: pre-election day (May 6th), in-election day (May 7th) and post-election day (May 8th). The number of mentions found were the following:

Party	Period	Mentions	Party	Period	Mentions
snp	pre-election	21668	green	pre-election	2631
labour	pre-election	11266	conservative	In-election	2002
snp	In-election	7087	green	In-election	1590
ukip	pre-election	5433	liberal	In-election	1154
conservative	pre-election	4368	labour	post-election	635
labour	In-election	4126	conservative	post-election	340
liberal	pre-election	3811	ukip	post-election	143
ukip	In-election	3498			

Figure 13: Number of mentions of each political party across 3 different time periods (days) in the general election

It was found that the Scottish National Party was by far the most mentioned party in the pre-election period, even though there were no candidates for the Birmingham area, perhaps through the fervour as a new political force, while Labour received far more mentions than the Conservative party or the Liberal Democrats. The numbers of post-election mention were significantly lower than other periods, suggesting that the election fervour dissipated very quickly in Birmingham once the election results were released. The following was the breakdown of the total sentiment score (summation) for each party and time period:

Party	Period	sentiment	Party	Period	sentiment
labour	pre-election	574.61	conservative	post-election	13.9
liberal	pre-election	378.56	liberal	In-election	13.15
green	In-election	247.1	green	post-election	11.8
ukip	In-election	226.84	ukip	post-election	10.3
green	pre-election	176	snp	post-election	5.3

labour	In-election	171.35	liberal	post-election	2.4
snp	pre-election	118.64	conservative	In-election	-36.84
snp	In-election	79.93	conservative	pre-election	-162.24
labour	post-election	41.73	ukip	pre-election	-185.09

Figure 14: Total sentiment for each political party, according to 3 time periods

It was observed that the Labour party was the most favoured political party according to the tweets harvested, leading the Liberal Democrats significantly by nearly 200 sentiment points (each tweet being scored from -1 to 1, and the total body of tweets being approximately 17,000). Other minor parties such as The Greens and the UK Independence Party was high in the ranking as well, leading to the knowledge that perhaps the Twitter population of Birmingham is more supportive/vocal of their support for these parties than the general population. These two parties did not win a seat in the Birmingham area, however the high positivity of sentiments is highly noteworthy, especially in the light of the view that the number of votes were not proportional to the number of seats won for political parties such as The Greens and UKIP[11].

Another notable observation was the stark contrast in the popularity of the Conservative party, compared with the national outcome of the elections. At the national level the election ended in an overall victory of the Conservative Party, gaining an outright majority in the lower house. However, according to the data collected, the total sentiment towards the Conservative party in all 3 time periods was in the bottom half of the ranking, with the pre-election sentiment being scored second to last.

This was reflective of the actual political sentiment in Birmingham, with pre-election polls[13] showing a prediction massively in favour of the Labour Party. This was also consistent with outcome of the election in the Birmingham area, where labour won 9 out of 10 seats in the actual elections[14].

Overall, our sentiment analysis was able to discover the most popular political party amongst the Twitter users in Birmingham, which was consistent with the political inclinations of the Birmingham population as a whole, and with the outcome of the general elections. Strong and vocal support for smaller parties were also found, being noteworthy as it was not visible by the election outcome alone.

Accent

It is common knowledge that one of the topics related to Birmingham is the distinctive 'Brummy accent' of Birmingham. Thus a scenario was devised, which attempted to analyse what people in Birmingham think about the Birmingham accent. The ‘topic_accent’ Map function was used to collect all tweets which mentioned 'accent' as a topic in the tweet. The concept list and entity list of each tweet, provided by the Meaningcloud service, was iterated over to look for the presence of 'accent', while the mention of the word 'accent' in itself was checked. Approximately 550 tweets were found satisfying

these conditions, and the average was computed using the 'accent_sentiment' function in conjunction with python processing.

The average sentiment score of these accents tweets were -0.031, which was significantly lower than, the average sentiment of all tweets in Birmingham, 0.2112 (computed with the 'birmingham_average_sentiment' function). The difference of over 0.2 out of a scale of -1 to 1 is striking. Examples of these accent tweets are the following:

TweetID 598823431874949120: “guy with broad brum accent trying to order five teas from polish lady in cafe. not going well.”

TweetID 594914712774713344: “The Birmingham accent is the worst accent I've ever come across in my life.”

TweetID 598746182815457280: “loved hearing the Brum accent and the locals are all so friendly unlike Londoners”

Figure 15: Example tweets mentioning the Birmingham accent

From these observations, it was surmised that there is a distinctive love-hate relationship between the people in Birmingham and their accent, with people expressing dislike more often than praise towards it. Through the calculation of sentiment scores, a degree of self-contempt was discovered, with the Birmingham accent considered to have a lower status amongst the other UK/English accents.

University

Another scenario covered was the life of the university students in Birmingham, whether there were any difference between the general population and the students in terms of the topics they talk about and the sentiments towards them. 6 views were created in total, all of which filtered the harvested tweets to find tweets with geo-location enabled, and those within the vicinity of:

i) University of Birmingham
ii) Birmingham City University
iii) Aston University
iv) Newman University, Birmingham
v) University College Birmingham

With 'university_average_sentiment' view, the tweets which had a sentiment score was gathered and the average calculated in conjunction with python processing. The average sentiment score was 0.177, which was slightly lower than the average of the entire city which was 0.212. This implied that university students were somewhat less happy than the general population, however it was also the case that the difference was not great enough to draw any definitive conclusion to this hypothesis.

With 'university_topics' view, the concepts and entities included in the Meaningcloud attributes of the tweets were emitted to compile a list of topics ordered according to the number of mentions:

Topic	Count	Senti.	Avg. senti.	Topic	Count	Senti.	Avg. senti.
#Birmingham	324	39.53	0.12201	#TweetMyJobs	14	10.9	0.77857
Birmingham	78	27.4	0.35128	attack	14	-7.9	-0.56429
#Job	56	41.36	0.73857	#Dudley	13	-2.5	-0.19231
#Jobs	51	37.76	0.74039	woman	13	-3.9	-0.3
United Kingdom	48	34.46	0.71792	beverage	12	8.22	0.685
#Hiring	33	24.75	0.75	job	12	7.17	0.5975
day	30	7.83	0.261	manager	12	7.1	0.59167
Hilton	27	18.86	0.69852	event	12	6.1	0.50833
#Hospitality	25	17.26	0.6904	@dermotodw	12	1	0.08333
man	24	-1.7	-0.07083	child	12	-0.8	-0.06667
food	21	11.3	0.5381	car	12	-2.8	-0.23333
home	21	-0.2	-0.00952	prison	12	-5.5	-0.45833
people	21	-1	-0.04762	#Transportation	11	8.1	0.73636
#RCPCH15	18	11.98	0.66556	canteen	11	6.6	0.6
West Midlands	18	8.3	0.46111	exam	11	-1.27	-0.11545
Wolverhampton	18	-0.01	-0.00556				

Figure 16: The top mentioned topics by University students in Birmingham (approximately 1000 out of 1 million tweets). The values, from the left, are the total number of mentions, the total sentiment score, and the average sentiment (the divided product of second and first value).

Out of these topics, the ones that drew our attention most was all of the Hash Tags related to jobs, and their unexpectedly high sentiment average. With Birmingham currently having more than double the unemployment rate compared with the national average[15], this was considered as a topic worth investigating further.

The view ‘university_jobs_tally’ and ‘university_jobs’ were created in order investigate the kinds of tweets being disseminated in the university areas and the kinds of jobs being offered around the university area. The aim of these MapReduce functions were to get a picture of the job situation from

the point of view of the students, and get an explanation on why the average sentiment was so positive despite the high unemployment rate.

What was found was that a large number of the job related tweets were in fact advertisements for jobs, where the tone of the advertisement were classified as positive by the Meaningcloud classifier. 191 tweets were extracted using this view, and here are some of the example tweets found:

["#Jobs", "#Healthcare #Job alert: CARE ASSISTANT in Edgbaston, B'ham | Sunrise Senior Living UK | #Edgbaston http://t.co/xvTL727nNM #Jobs #Hiring", 0.8]

["#Hiring", "Hays #BusinessMgmt #Job: Project Manager- M&E (#Birmingham, UK) http://t.co/a3VqJNOlq6 #Jobs #Hiring #TweetMyJobs", 0.8]

["#Hiring", "Resourcing Solutions #Transportation #Job: Possession/Train Planner (#Birmingham, UK) http://t.co/v00VBu3o3G #Jobs #Hiring #TweetMyJobs", 0.7]

Figure 17: Example job advertisement tweets with their sentiment score value

In addition, the tally of the types of job being offered in the university area was the following:

Job	**Mentions**	**Score**	**Job**	**Mentions**	**Score**
surveyor	7	5.4	receptionist	2	1.4
assistant	6	4.3	supervisor	2	1.4
manager	5	3.9	planner	2	1.4
consultant	3	2.4	engineer	1	0.8
director	2	1.6	chef	1	0.7

Figure 18: tally of jobs, with the first value the frequency of occurrence and the second the sentiment total value

This tally indicates a variety of jobs being offered for different types of skillsets, from white collar to blue collar. However, not enough tweets were harvested to have a comprehensive list of jobs to decipher which ones are especially high in demand.

The only conclusion drawn from this scenario was that the high sentiment average was explained by the large number of job advertisements. Perhaps this means that the prevalence of these advertisements in itself reflect the bleaker youth employment situation in Birmingham.

Sports, Aston Villa Football Club

Football, or soccer, is one of the most popular sports in Birmingham and the UK, and it is an aspect of life which was deemed important to be covered in the scenarios. The Aston Villa Football Club was chosen as the target for this scenario as the biggest club in Birmingham. The views 'most_mentioned_avfc_players' and 'most_positive_avfc_player' attempted to capture what people in Birmingham thought of each player in the club. Using the map function, tweets with the Hashtag #avfc were extracted, where the tweet was checked if it contained the player names as well as the sentiment score of the tweets. The following were the results:

Position	Player	mentions	Position	Player	mentions
Coach	sherwood	219	GoalKeeper	guzan	15
Striker	benteke	59	GoalKeeper	given	15
Middlefielder	delph	33	Middlefielder	cole	14
Striker	grealish	30	defender	bacuna	12
defender	vlaar	26	Middlefielder	gil	9

Figure 19: The number of mentions of each Aston Villa FC player and coach, out of 505 tweets which contained the #avfc Hash Tag

Position	Player	Score	Position	Player	Score
Coach	sherwood	27.15	Middlefielder	westwood	3.12
Striker	benteke	15.4	Striker	grealish	2.9
Middlefielder	delph	10	GoalKeeper	given	1.9
Middlefielder	cole	3.6	defender	richardson	1.2
defender	bacuna	3.47	Middlefielder	gil	1.2

Figure 20: The total sentiment score of AVFC football players and coach, out of 185 tweets which contained a sentiment score

It was observed that the coach Sherwood was the most mentioned person in AVFC, who also received the highest total sentiment score. We hypothesised that it was usually the case that the striker of a team is the most popular player, and this was consistent with the result, with Benteke receiving 59 mentions and coming in second. This was followed by Delph, Grealish, Vlaar and Guzan. It was also shown that all Goalkeepers of the club had a relatively low or negative score, with Given just making to the top list as 8th.

Happiest hours of the Week/When Birmingham people tweet the most

In uncovering the life of Birmingham, a key question which presented itself was the general tweeting behaviour of the population. A scenario was devised to answer the following questions: whether the average level of positivity across all tweets changed according to the time of the week, and whether there was a difference in the frequency of tweets depending on the time and day. With the assumption that the tweeting population reflected the general populace of Birmingham, it was hoped that finding a pattern will lead to an induction that would provide a knowledge about the Birmingham people's daily behaviour. The views 'most_frequent_tweet_hour' and 'sentiment_morning_night' were implemented to answer the questions posed by this scenario. The former computed the percentage of tweets for each hour, relative to the total number of tweets harvested for that weekday, while the latter worked out the average sentiment for the weekday and time period, according to the total number of tweets in that specific time period. The following were the results:

Day, Hour	percentage	Day, Hour	percentage
Thursday,7	19.40%	Wednesday,18	8.18%
Thursday,8	19.28%	Wednesday,17	7.60%
Thursday,6	15.78%	Tuesday,14	7.36%
Wednesday,21	12.55%	Friday,16	7.23%
Friday,18	12.19%	Saturday,14	7.23%
Wednesday,20	10.67%	Monday,20	7.19%
Wednesday,19	9.09%	Tuesday,12	7.17%
Wednesday,22	8.58%		

Figure 21: Hours of the week where there were the highest ratio of tweets relative to the total of that day, with the hour in 24 hour format

Period	score	Period	score
Tuesday Morning	0.31394	Sunday Morning	0.25512
Sunday Afternoon	0.30589	Wednesday Morning	0.25445
Friday Morning	0.30181	Monday Night	0.2449
Saturday Afternoon	0.29583	Saturday Morning	0.24088

Tuesday Afternoon	0.2891	Wednesday Night	0.23646
Thursday Morning	0.28453	Monday Morning	0.21481
Thursday Night	0.28225	Monday Afternoon	0.20848
Thursday Afternoon	0.2812	Saturday Night	0.2015
Tuesday Night	0.27773	Wednesday Afternoon	0.13952
Friday Afternoon	0.2741	Friday Night	-0.06174
Sunday Night	0.26115	Sunday Morning	0.25512

Figure 22: Average sentiment score of each weekday and time period, with morning being 7am-12pm, afternoon 1pm-6pm and night 7pm-12am

From the collected data it was found that the Birmingham population tweeted the most in the Thursday morning period, with Wednesday at 9pm ranked next followed by Friday 6pm. It was surmised that perhaps people were relatively unoccupied at these times of the week, while more busy during weekend nights.

Another surprising discovery was that the highest average sentiment periods were found to be in the after morning hours on Thursdays, with Tuesday afternoon and night coming in third and fourth. This was not in line with the original hypothesis, that people would be the happiest on the weekend, away from the shackles of life or study.

Although it is difficult to come to any definitive conclusions, Perhaps the social pattern is that the people of Birmingham are too busy with their lives to tweet their sentiments during the peak weekend hours, where instead the tweets about the anticipation of what is coming up on the weekend, posted before the weekend on Thursdays, dominate in the tally of positive tweets. It was also surmised perhaps Tuesdays are when people readjust to the regular cycle of study/work, where they find a temporary period of higher positivity after the negative transition period on Mondays, right after the weekend.

Language of the tweets

Another facet of Birmingham life which was investigated was the multicultural composition of the city. As Melbourne is a multicultural city, it was deemed that looking at the languages of the harvested tweets was a viable approach. In order to do this, the view 'user_tweet_language' was created to correlate the languages of the Twitter accounts. The following is the top 20 ranking of the languages, according to the user setting of the Twitter account:

Language	count	Language	count	Language	count	Language	count
en	793457	id	1545	fi	136	gl	24
en-gb	84202	ja	1530	ko	131	en-au	23
es	13833	nl	1372	he	122	hi	23
fr	8167	tr	1192	zh-cn	109	vi	19
pt	6005	ar	723	hu	107	fil	19
ru	3895	sv	394	no	81	eu	14
it	3578	da	304	cs	81	ga	11
select language	2301	th	270	xx-lc	70	nb	10
de	2163	ca	255	zh-tw	55	zh-hans	10
pl	1568	el	147	ro	53		

Figure 23: Ranking of the user language of the harvested tweets

Without a surprise, English was the overwhelming majority out of all the languages of the Twitter accounts, with over 90% of the total number of tweets. All of the other languages except Spanish were less than 1% of the total number of tweets. Other European languages were prominent in the higher end of the rankings, with French, Portuguese, Russian and Italian, German and Polish in the top 10. This presents us with the knowledge that there exists small pockets of multiple minorities in Birmingham from countries across Europe.

The most prevalent Asian language was Indonesian (id, 1545), with Japanese as the close second (ja, 1530). This present a picture of Birmingham as a popular destination for travel, exchange study or immigration. There were 23 accounts registered in Australian English as well (en-au 23).

By reducing the group level to 2, it was also found that many Twitter users tweeted in a language different to the language they registered their accounts in. The most prominent user/tweet language combination was Spanish Twitter accounts tweeting in English. Many other unexpected combinations were found as well in small numbers, including a Japanese Twitter account tweeting in Arabic. These different language compositions indicate the cosmopolitan nature of the city, with many multilingual individuals using Twitter to express their messages in a multitude of languages.

Most followed Twitter users in Birmingham

Another scenario covered was the most influential people in Birmingham. This scenario was conceptualised as it was considered important to find any vocal people who are local to Birmingham, as

they may lead to the discovery of aspects of life specific to Birmingham. To do this, a spatial filter was applied to all of the harvested tweets, and the Twitter accounts with the most number of followers were searched out of these accounts. The view 'most_followers' contained these results:

username	followers	username	followers
@boohoo	394267	@AmarnaMiller	52083
@UlissesWorld	135287	@davidgcant	46858
@twinatlantic	80982	@craigbigbro1	46676
@JsGravendijk	68280	@mac_birmingham	39028
@WWE__History	64555	@JeromeShaw	38324

Out of these Top 10, only @boohoo, @davidgcant, @craigbigbro1 and @mac_birmingham were accounts from the UK. Thus, with this search, influential local figures were not discovered at all, with @davidgcant and @mac_birmingham being the only accounts local to the Birmingham region. @davidgcant is a notable 'Chartered Construction Safety and Health Practitioner' (http://www.veritas-consulting.co.uk/about_david_cant_veritas_office), while @mac_birmingham is an account for a local Arts/Cultural Centre in Birmingham, with over 920,000 visitors per year.

Instead, this map function pointed to a scenario which showed some of the public figures who were travellers in Birmingham, including @JeromeShaw, a well known travel writer from Denver, USA and @JsGravendijk, an entrepeneur/CEO from Rotterdam, Netherlands. It also included some businesses that were successful in Birmingham, such as @boohoo, a global fashion retailer.

Overall the tweets with the most followers in Birmingham contained mainly individuals or businesses not head quartered in Birmingham, revealing some of the external elements that influence/visit the city.

Conclusion

In summary, we created a series of services and programs working together as an integrated system, and they were used to harvest 1,017,844 tweets from Birmingham from late April to mid May 2015. The team utilised the topic extraction and the classification of sentiments (positive, neutral and negative) provided by Meaningcloud, a third-partyservice. A series of MapReduce functions were created corresponding to the various scenarios that were devised, in order to capture some of the aspects of life that exists in Birmingham. These include the General Elections of 2015, recent concerts by the Vamps, sports events, and university students lifestyle. We concluded from the data that: Birmingham residents identified with the Labour party the most during the elections 2015 (200 points) ,

while having vocal support for the smaller political parties such as The Greens and UKIP, that they are not particularly proud of their English accent (- 0.2 rate), that the university student's second most mentioned topic was job hunting (56 mentions), possibly due to the unemployment situation in Birmingham, that the coach of the local football team Aston Villa was the most mentioned member of the team (219 mentions). From the views created with the MapReduce functions, we also observed that in Birmingham the highest average sentiment time period of Tweets were posted on Tuesday mornings (0.28453 rate), with the highest ratio of tweets being posted on Thursday mornings. Spanish was the second biggest denomination after English in the variety of languages found to be used and registered for the Twitter accounts used in Birmingham (1.35%). Through the entirety of the process of this project, the team was able to learn the technical details and the workings of the NeCTAR Research Cloud, Ansible, couchDB, MapReduce and the difficult task of working with a great amount of data in order to transfer raw data into palpable knowledge about the social world.

References

[1] Twitter website. Retrieved May 19, 2015, from https://about.twitter.com/company

[2] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107-113.

[3] (2015). MeaningCloud: Web Services for Text Analytics and Mining. Retrieved May 19, 2015, from https://www.meaningcloud.com/.

[4] (2013). Jax-RS - Java.net. Retrieved May 19, 2015, from https://jax-rs-spec.java.net/.

[5] (2010). Twitter Developers. Retrieved May 18, 2015, from https://dev.twitter.com/.

[6] (2014). Tweepy. Retrieved May 18, 2015, from http://www.tweepy.org/.

[7](2014) Bounding Box Tool: Metadata Enrichment for Catalogue ... Retrieved May 18, 2015, from http://boundingbox.klokantech.com/.

[8](2015). Sentiment analysis and opinion mining API | MeaningCloud. Retrieved May 19, 2015, from http://www.meaningcloud.com/products/sentiment-analysis/.

[9]MVCC means no locking. Retrieved May 18, 2015 from https://people.apache.org/~dch/snapshots/couchdb/20140715/new-docs-build/html/intro/consistency.html

[10] Retrieved May 19, 2015 from https://www.nectar.org.au/about-nectar

[11] (2006). AWS | Amazon Elastic Compute Cloud (EC2) - Scalable ... Retrieved May 19, 2015, from http://aws.amazon.com/ec2/.

[12](2015). Election 2015: What difference would Would Proportional Representation Have Made? - BBC.com. Retrieved May 17, 2015, from http://www.bbc.com/news/election-2015-32601281.

[13](2015). Final General Election Predictions 64: Birmingham - Iain Dale. Retrieved May 17, 2015, from http://www.iaindale.com/posts/2015/05/05/final-general-election-predictions-64-birmingham.

[14](2015). 2015 General Election Results - Birmingham City Council. Retrieved May 17, 2015, from http://www.birmingham.gov.uk/2015-General-Election-Results.

[15] (2015). Unemployment Briefing - May 2015 - Birmingham City Council. Retrieved May 19, 2015, from http://www.birmingham.gov.uk/cs/Satellite?blobcol=urldata&blobheader=application%2Fpdf&blobheadername1=Content-Disposition&blobkey=id&blobtable=MungoBlobs&blobwhere=1223585952793&ssbinary=true&blobheadervalue1=attachment%3B+filename%3D958167Unemployment_Briefing_May_2015.pdf.

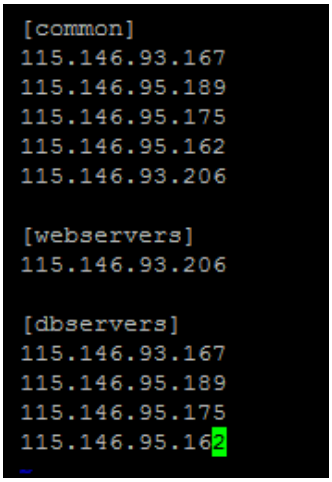
Appendix/User Guide

GitHub Repository containing all code and JAR files of the system:
https://github.com/matheumalo/twitter_project

Ansible and Setup

For the deployment of the necessary software environment we came with the idea of define 3 main roles. One role is called “common” which installs basic software such as python, java, update of some packages and some libraries. The second role is called “install-dbservers” which installs the CouchDB database, install tweepy, and the twitterHarvest service. The last role is called “install-webservers” which installs the Apache tomcat web server and the executable frontend .WAR

- First copy our ansible folder (ALL files) (which contains roles, playbooks and hosts) into the /ansible directory.
- Second, modify the hosts file. Add the IP address of the instance under the desire role (the instance could play many roles). i.e



Check our main playbook “install.yml” and confirm that the hosts is correct.

- Third execute: ansible-playbook --private-key=[private key] install.yml

Using CouchDB and adding MapReduce functions to DB

The python program '/map_reduce_search_python.py' is the program used to create read or update views from the couchDB database of harvested tweets. It uses functions implemented inside '/couchdb_processor.py' to perform these operations.

To add new views to the database to perform a MapReduce search:

1. Add the JavaScript function in the folder /map_reduce_function/
2. Inside '/map_reduce_search_python.py', create a new view with couchdb.create_view(), with the new Map function and reduce Function as parameter, as well as the db name and location
3. Store the dictionary object returned by couchdb.create_view() into a variable, and use that as the first parameter for calling couchdb.sort_map_reduce_search(). This function prints out the view, with value based sorting in descending order. Specify the Top N items you want returned as the second parameter, with 0 for all documents in the view. Third parameter is for the group level inside couchDB, and the fourth is the mode for any additional aggregation operations performed in Python. 'avg' mode will return list[1]/list[0] for a mean calculation after the '_sum' reduce operation, and so on.