

Faculty of Technology, Design and Environment

Department of Computing & Communication Technologies

BSc (Single Honours) Degree Project

Title:

Sentiment Analysis on Real-Time Social Media Data

Surname:	Sykes
First Name:	Alicia
Student Number:	12011471
Number Module:	U08096
Supervisor:	David Lightfoot
Second Supervisor:	Clare Martin
Date Submitted:	14th April 2016

Extract from the Student Conduct Regulations:

2.2.1 Students shall not cheat (ie obtain, or attempt to obtain, an unfair academic advantage) in any assessment of their competencies or academic ability or professional skills. They shall comply at all times with the provisions of the Regulations for Students taking Assessments. In particular, they shall not commit collusion, plagiarism, falsification, duplication, submit the work of others as their own or allow another person to undertake an assessment for them.

Explanation of terms used in the Student Conduct Regulations:

Impersonation means taking an assessment on behalf of another student, or allowing another person to take an assessment on your behalf.

Collusion means producing assessed work by working with another person whom you have not been authorised to work with by the Module Leader. This includes, but is not limited to, allowing another student to copy your work.

Falsification means presenting invented data, for example claiming that a program works when it does not, or claiming that it produces results which it actually does not.

Plagiarism means submitting the work of someone else as if it were your own. If you include material by someone else in your assignment, you must show clearly in the text how much was copied from elsewhere. It is not enough just to list references at the end of your assignment.

Guidance on the correct use of references can be found on www.brookes.ac.uk/services/library, and also in a handout in the Library.

If you do not understand what any of these terms mean, you should ask your Module Leader to clarify them for you. The full regulations may be read in the Library, or accessed on-line at <http://www.brookes.ac.uk/regulations/stureqs.html>

If you do not understand what any of these terms mean, you should ask your Project Supervisor to clarify them for you.

I declare that I have read and understood Regulations 2.2.1 of the Regulations governing Academic Misconduct, and that the work I submit is fully in accordance with them.

Signature..... Date.....

**REGULATIONS GOVERNING THE DEPOSIT AND USE OF OXFORD BROOKES UNIVERSITY
MODULAR PROGRAMME PROJECTS AND DISSERTATIONS**

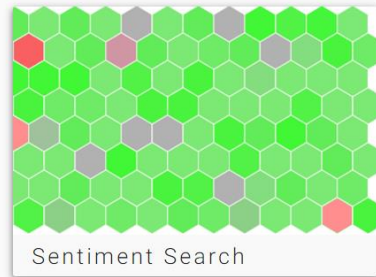
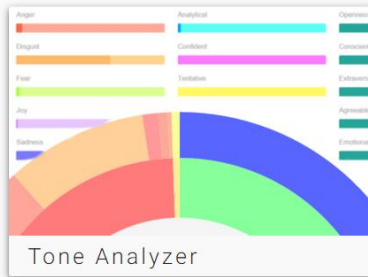
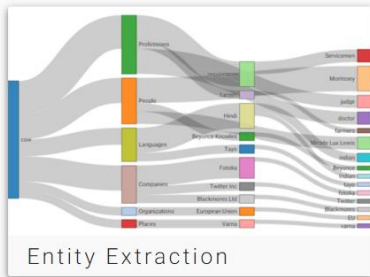
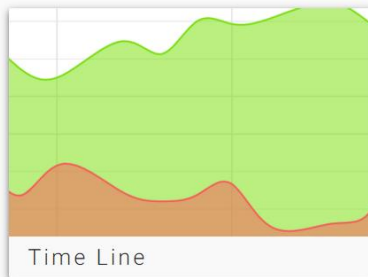
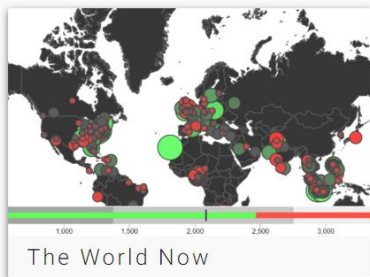
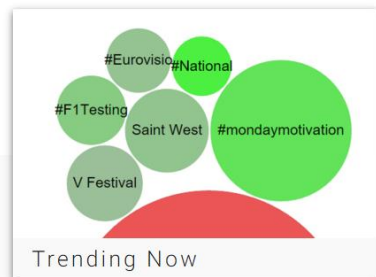
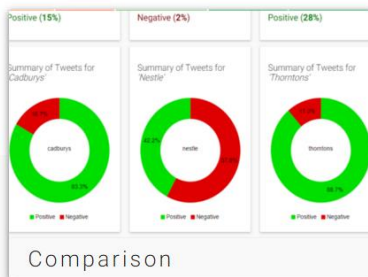
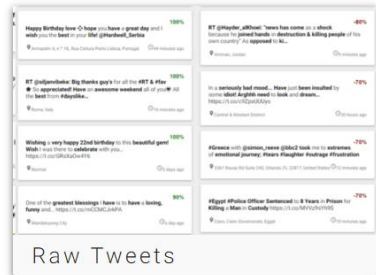
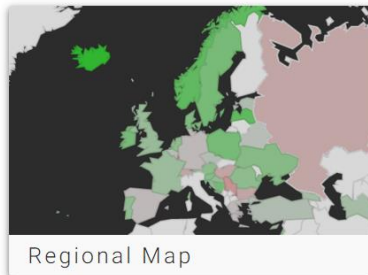
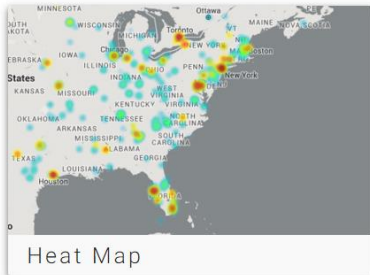
Copies of projects/dissertations, submitted in fulfilment of Modular Programme requirements and achieving marks of 60% or above, shall normally be kept by the Library.

I agree that this dissertation may be available for reading and photocopying in accordance with the Regulations governing use of the Library.

Signature..... Date.....

Sentiment Analysis on Real-Time Social Media Data

The research and development of a sentiment analysis module, and the implementation of it on real-time social media data, to generate a series of live visual representations of sentiment towards a specific topic or by location in order to find trends.



Live Demo of Final Application: <http://sentiment-sweep.com>



Documented Code, Published Opensource: <https://git.io/vVhGy>

Alicia Sykes
April 2016

ABSTRACT

The research and development of an artificial intelligent, sentiment analysis module, which detects average moods and attitudes from a string of text. It is then applied to a continuous stream of real-time Twitter data. A web application renders these live sentiment results in the form of a series of data visualisations that allow trends to be plotted between people's attitudes and other factors, such as location or time.

The solution is primarily written in CoffeeScript, with a Node.js backend, and an Isomorphic frontend. It is unit tested and all modules developed are documented and published to the opensource community.

ACKNOWLEDGEMENTS

I would like to thank my supervisor David Lightfoot, and my module leader Chris Cox for their helpful resources and feedback during the duration of the project.

I would also like to thank everyone who has ever tweeted, this project wouldn't have been possible without the data made available by the general public on Twitter.

Nor would it have been possible for me to develop anything near this scale without the tremendous amount of open source resources, made freely available online by other developers in the JavaScript community.

I would like to thank the good people on Stack Overflow for posting and maintaining an extensive bank of programming related questions and answers, without which I might not have got much further than a Hello World JavaScript application.

Thank you to all the participants who took part in the research surveys as part of this application.

Accenture very kindly gave me time and resources to make a start on this project while I was on an extended placement with them.

Thank you to Jesus, for the miracle that this quite ambitious project was completed within the time frame and works as expected, producing correct results.

Finally, I'd like to thank my friends for putting up with me talking about nothing except how cool sentiment analysis is, over the past couple of months. Sentiment Analysis really is awesome, as I hope this project demonstrates.

TABLE OF CONTENTS

Table of Contents.....	5
1 Introduction.....	7
1.1 The Problem	7
1.2 Project Aims.....	7
1.3 Brief Description of Proposed Deliverable	8
1.4 Relevance to Computer Science BSc Degree Program	8
1.5 Report Structure	8
2 Background Research.....	9
2.1 Sentiment Analysis: An executive assessment of representing trends in opinions expressed on social media	9
2.2 Sentiment Analysis: Comparing Technical Approaches	11
3 Methodology	15
3.1 Development Plan.....	15
3.2 Development Tools	21
3.3 Testing.....	24
3.4 Code Style and In-Code Documentation	27
3.5 Front-end Plan	28
3.6 Backend Plan	36
4 Implementation	41
4.1 Sprint 0 – Project setup	41
4.2 Sprint 1 – Create base application and database schema	49
4.3 Sprint 2 – All tweet handling backend modules	51
4.4 Sprint 3 – All location and utility backend modules	54
4.5 Sprint 4 – Sentiment Analysis module	55
4.6 Sprint 5 – Integrate backend modules into application	56
4.7 Sprint 6 – Geo sentiment data visualisations	57
4.8 Sprint 7 – Keyword sentiment data visualisations.....	61
4.9 Sprint 8 – Timeline, trending and comparison charts	65
4.10 Sprint 9 – Real-time Functionality	71
4.11 Sprint 10 – Advanced Data Visualisations.....	72
4.12 Sprint 11 – Home page and search page.....	74
4.13 Sprint 12 – Finishing, UX Testing and Launching	80

5	Evaluation	84
5.1	Evaluation of Requirements Met	84
5.2	User Experience Evaluation	90
5.3	Time Management and Scheduling	90
5.4	Evaluation of Technical Testing.....	91
5.5	Evaluation of Code Quality	91
5.6	Evaluation of Addressing Legal, Social and Ethical Issues	92
5.7	Self-Evaluation	93
5.8	Limitations of Existing Solution	94
5.9	Further Work and Future Enhancements.....	94
5.10	Example Findings using the Final Application	95
6	Conclusion	96
6.1	Aim.....	96
6.2	Final Stats	96
6.3	Link to Final Solution	96
7	References	97
7.1	Research References	97
7.2	Methodology References.....	98

Appendix

1	Appendix One - User Stories	102
2	Appendix Two - Style Guides	130
3	Appendix Three - VCS History	132
4	Appendix Four - Test Result Examples	150
5	Appendix Five - User Experience Survey	161
6	Appendix Six - Code Listing	165
7	Appendix Seven - Tech Stack	296

1 INTRODUCTION

1.1 THE PROBLEM

Too much data, no way to clearly visualise overall trend

There is a deluge of social media data available every day. Too much data for swift accurate processing by people to determine the overall message conveyed.

Brands need to understand the overall feeling towards their new products. They could read 2,000 tweets - but it would not be a good use of time. Instead a better solution would be to use a sentiment analysis AI module to analyse and show overall feelings towards their product.

Alternatibley if you're in a theme park and you would see which rides or parts of the park are having positive reactions. Reading through thousands of tweets relating for each ride or area would take a while. Instead a visualisation could show where positive conversations or tweets are happening as opposed to less positive reactions.

There are many other times you may want to quickly gauge overall sentiment towards a topic, such as comparing opinions on two politicians running in an election, or what particular trends of time and topic relate to each other.

1.2 PROJECT AIM

The aim of this project is to develop and publish an open source sentiment analysis package. In conjunction to this, a web platform will be created, which uses the package to make the opinions conveyed on social media quantifiable and represented in a clear visual format. It will allow for trends to be found between sentiment and other factors.

1.3 PROJECT OBJECTIVES

The objective of this project are:

- Fetch relevant results from the mass amount of social media data available
- Automatically calculate opinions on this data set, to allow for users to gauge overall attitude towards a given topic without having to read each tweet or post
- Plot a series of visual analytics, to find and display trends between the sentiment data and other factors such as time, location, keywords or other topics
- To have a single centralised dashboard providing all useful data to the user in real-time and with links to further break down the results

A secondary objective of the application is to research into the area of sentiment analysis, and to develop and publish an open sauce sentiment analysis module.

All objectives will be measured based on meeting acceptance criteria, unit testing and user experience testing. This is outlined in the methodology.

1.4 BRIEF DESCRIPTION OF PROPOSED DELIVERABLE

The proposed deliverable will be in two parts:

- A web application, publicly accessible through any modern browser. It will contain links to each of the dynamic real-time sentiment visualisations, as well as a search page where the user can enter a custom topic, and fresh tweets will be fetched, analysed and rendered.
- A series of open source packages to do specific tasks (including the sentiment-analysis module, a fetch-tweets module, geo-lookup etc....) Each of these will be tested, documented and then publicly published for other developers to make use of.

1.5 RELEVANCE TO COMPUTER SCIENCE BSC DEGREE PROGRAM

This project is based around five main topics within the field of computer science.

- **Visual analytics** - The final solution will need to present data in the form of a series of dynamic and interactive data visualisations. This will include researching the most appropriate charts and graphs to be utilised as well as which technologies to use
- **Big data** – The solution will be required to work with very large datasets, in order to portray accurate overall results. This use of big data, will, of course, mean the algorithms are efficient and the application is thoroughly tested
- **Social Media** – All the data will come from social media sources (primarily Twitter). A knowledge of social media and how best to extract, analyse and display results will be necessary
- **Artificial Intelligence** – In order to implement some of the most advanced functionality, an AI engine (IBM Watson) will be utilised. A simplified machine learning algorithm will also be developed, and used to improve the accuracy of results over time
- **User Experience (UX)** – It will be essential that the final results are presented in a clean and concise way, to allow users to quickly interpret the data with little prior knowledge

1.6 REPORT STRUCTURE

1.6.1 Final Report

1. **Introduction** (this section) – contains a brief outline to the project that will be developed, and an insight into the market, and the current solutions already available
2. **Background Research** – background research in the form of literature reviews, surveys and several experiments, carried out to gain a better insight into the solutions requirements and proposed technical approaches
3. **Methodology** – detailed plan outlining exactly how the all aspects of the solution will be developed and launched from a technical point of view
4. **Implementation** – how the system was developed, including annotated screenshots and code snippets from the final version, and next steps
5. **Evaluation** – review of the final solution, including findings, user survey results, and a write up of degree of success
6. **References** – Information sources used in the research and development of the final solution

1.6.2 Appendix

1. **Appendix 1** – User stories
2. **Appendix 2** – Style guides
3. **Appendix 3** – Version Control
4. **Appendix 4** – Set of test results
5. **Appendix 6** – User Experience Survey
6. **Appendix 6** – Code Listing Sample
7. **Appendix 7** – Tech Stack Summary

2 BACKGROUND RESEARCH

2.1 SENTIMENT ANALYSIS:

AN EXECUTIVE ASSESSMENT OF REPRESENTING TRENDS IN OPINIONS EXPRESSED ON SOCIAL MEDIA

2.1.1 Abstract

The purpose of this literature review is to explore how sentiment analysis can be used to visually represent people's opinions, and more specifically how these visual representations can illustrate trends of sentiment mapped against factors such as time, location or topic. It will also touch on the different methods of analysing sentiment as this dictates what type of trends can be mapped.

2.1.2 Introduction

Despite the mass amounts of data uploaded every day to public social media networks (500 million Tweets per day), and the advances in sentiment analysis and natural language understanding. Research into applying sentiment analysis to social media data to find trends has been limited. The purpose of this review is to explore the potential of gaining an insight into an overall attitude towards specific topics from analysing social media channels, and how the trends from these attitudes can have practical applications.

2.1.3 Related Work and Current Applications for Twitter Data

Twitter data can be used to show trends and then make predictions for the future based on historical events. Meesad (2014) outlines how trends from past events can be mapped to stock prices, and this, in turn, can be used to reasonably accurately predict future stock prices, aiding investors to make better trading decisions.

The social Web is also being commercially exploited for purposes such as automatically extracting customer opinions about products or brands (Bansal et. Al 2004). This gleaning of Twitter data can then be used to gain a deeper understanding of people's behaviour and actions (Wenbo Wang et. al 2012). One key use for this insight into people opinions would be to aid marketing campaigns, as companies will have a better understanding of what techniques were effective in successfully marketing a product or service.

Dr. Tariq Mahmood et. Al (2013) describes how in the 2013 Pakistani Elections, they were able to use a large set of Twitter data from Pakistan, to accurately predict the winners of the election. The algorithm worked by applying a series of predictive rules to the data and categorising Tweets based on which rules they followed. This again provides a valuable insight into the country's political future before any official results have been released, and information like this couldn't possibly be collected on this scale with any conventional data collection method.

2.1.4 Comparing the different methods of sentiment analysis

There are several different ways of calculating the sentiment from a string of text, but two key underlying approaches. The first is a dictionary based method, this is where there is a database of words each with a score of how positive or negative the word is. An algorithm then calculates the overall aggregated score for a given string based on the database. The second method uses natural language understanding and machine language to return much more detailed and accurate results.

However, this is not readily available without large amounts of computing power nor is it easily possible on large sets of data.

Within the machine language method, there are a number of different algorithms that can be used. Dr. Tariq Mahmood et. Al (2013) shows a good comparison between three of them on page 4 of his paper. The CHAID algorithm was slightly more accurate than the Naïve Bayes and Support Vector Machine but the results were close. These algorithms are used as the base for the data structure which in turn dictates what order it gets sorted, which will directly affect results as the system learns as it goes along. This works in a similar way to the semantic text plagiarism detection technique outlined by Osman (2013) which looks at semantic allocations of each sentence to gain an understanding of what the underlying message is.

Vu Dung Nguyen (2013) quantitatively compares both the dictionary-based and the machine learning approach to sentiment analysis. In this paper, they were studying reactions to the Royal birth.

A key difference between the efficiency of the two approaches is that the machine learning method requires each Tweet to be analysed individually and then the results are aggregated, however the dictionary-based approach can analyse all Tweets at once, as it is simply assigning a score by each word. The following diagram illustrates this concept.

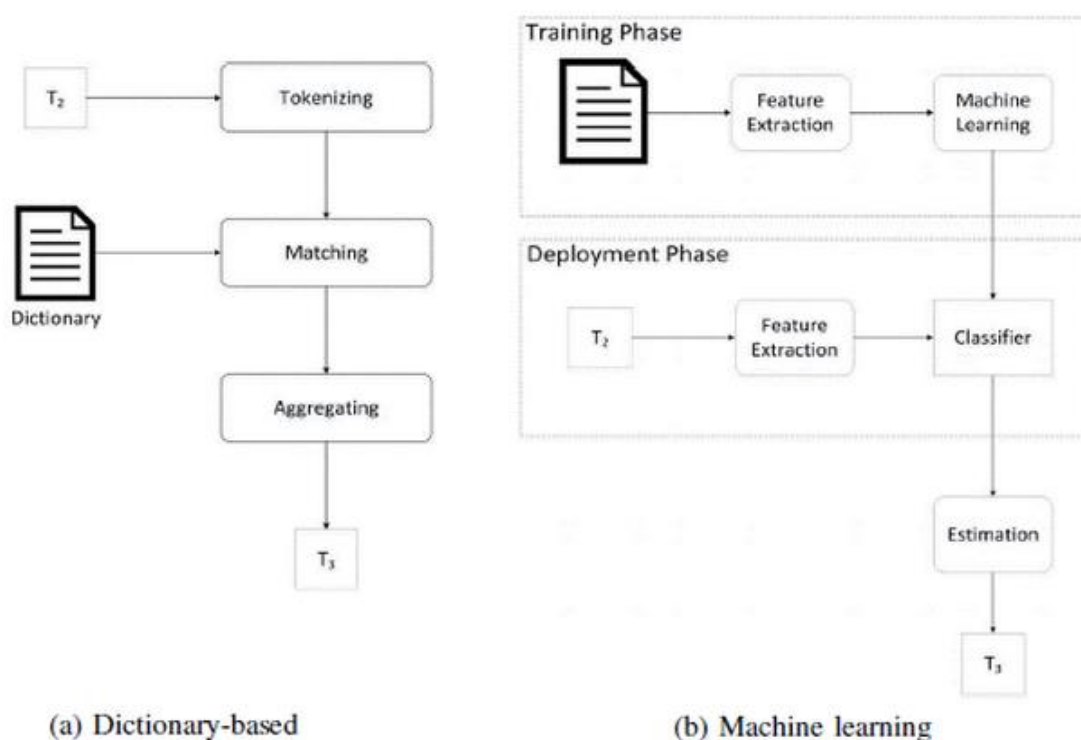


Fig 1 Dictionary Vs Machine learning SA

When comparing the results of dictionary-based methods and machine language methods, there are some differences. The dictionary-based results seem to be consistently lower (more negative) than the natural language results, however, the overall sentiment trend that is plotted is very consistent between the two methods. This is the difference in sentiment value is likely to be caused by there being significantly more negative words in the English language than positive words.

2.2 SENTIMENT ANALYSIS: COMPARING TECHNICAL APPROACHES

2.2.1 What is Sentiment Analysis?

Sentiment analysis is the process of computationally identifying opinions expressed in a piece of text, to determine the overall attitude conveyed. At its most basic level, this could be resolving the string into an integer score that represents positivity. It can, however, go a lot further, and identify keywords in the text and then compute what the authors feelings and attitudes are towards that topic. The results are of course just subjective impressions and not facts, but with large sets of data can build up a very accurate representation of people's opinions.

There is a growing demand for SA to make sense of a large amount of data representing people's opinions. It can be used to understand attitudes conveyed in mass amounts of Twitter data, or to analyse product reviews or to categorise customer emails, to name just a few of its applications.

The purpose of this paper is to carry out some quantitative and qualitative research compares different readily-available methods of SA. The two most common SA methods are dictionary based and natural language understanding based model. As a benchmark the results from both of these will also be compared with human computed values, which are likely to be much more accurate although considerably slower to compute.

2.2.2 Dictionary-Based Sentiment Analysis

The lexicon-based approach involves calculating orientation for a document from the semantic orientation of words or phrases in the document (Turney 2002). This is usually done with a predefined dataset of words annotated with their semantic values, and a simple algorithm can then calculate an overall semantic score for a given string. Dictionaries for this approach can either be created manually (see also Stone et al. 1966; Tong 2001), or automatically, using seed words to expand the list of words (Hatzivassiloglou and McKeown 1997; Turney 2002; Turney and Littman 2003).

2.2.3 Natural Language Understanding Approach

The natural language understanding (NLU) or text classification approach involves building classifiers from labelled instances of texts or sentences (Pang, Lee, and Vaithyanathan 2002), essentially a supervised classification task. There are various NLU algorithms, the two main branches are supervised and unsupervised machine learning. A supervised learning algorithm generally builds a classification model on a large annotated corpus. Its accuracy is mainly based on the quality of the annotation, and usually the training process will take a long time. Unsupervised uses a sentiment dictionary, rather like the lexicon-based approach, with the addition that builds up a database of common phrases and their aggregated sentiment as well.

2.2.4 Research Plan Methodology

As part of the research, I am going to conduct a research experiment to compare the results produced by natural language understanding (NLU) SA methods to the dictionary based SA approach. There will also be a set of results computed by a human to be used as a benchmark.

The natural language understanding component

The HP Haven OnDemand API is a powerful natural language understanding engine, and will be used for the NLU component. It is free to use for a limited number of requests and provides more details than just an aggregate sentiment score. I have developed a custom wrapper module for this experiment, and the source code and documentation for it can be viewed at <https://goo.gl/NTXfyp>

The dictionary-based component

I have developed a simple dictionary-based algorithm, and have packaged it up as a standalone module, this will be used for the dictionary-based component. For the dataset, it will make use of the AFINN-111-word list, which is a comprehensive list of English words annotated with an integer for valence. The source code and documentation can be viewed at <https://goo.gl/gU4f9A>

The human component

To provide a basic benchmark for results, a survey including a sample of Tweets that will be analysed by the two systems will be drawn up. Participants of the research will be asked to rate each Tweet with a score between 0 (very negative) and 10 (very positive) with 5 being neutral. Since the survey will be considerably more time-consuming than the other two methods, only a sample of the data will be analysed by the five participants.

Data source

The data source will be Tweets regarding the Edward Snowden case in 2013. The Twitter API will be used to supply the Tweets. I have written and published a custom module to facilitate the easy fetching of relevant Tweets, see <https://goo.gl/WQy7gl> for source code and documentation.

Rendering and Displaying Results

Since the results will be dynamic (able to change if a different query is passed in), the charts must be flexible. A combination of Google Charts JavaScript library and a custom module written in D3.js will be used.

To view the final solution online, and check out the comparison tool for yourself, visit:

<http://sentiment-sweep.com/sa-comparison>

There are also links to the documentation from here, and a brief explanation of how it works.

2.2.5 The Result

Description of Graph

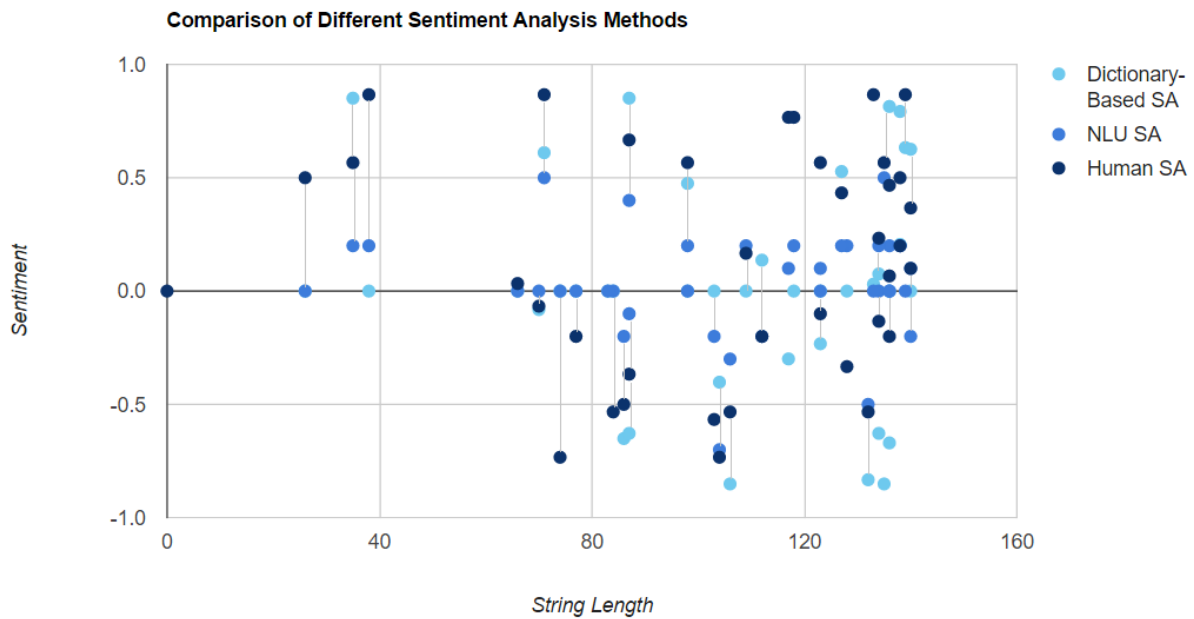


Fig 2 Scatter chart showing accuracy of various SA methods

Each connected point (three or fewer dots, connected with a single vertical line) represents a Tweet. Where each dot is calculated sentiment analysis result, the light blue is dictionary based results, mid-blue is NLU-based results and the dark blue are the benchmark results calculated by humans in the survey. The lines between the points indicate that they were generated from the same Tweet. Some points do not have lines because the dictionary result was exactly the same as the NLU result. The x-axis shows Tweet length (i.e. string length between 0 - 160). The y-axis a measure of overall sentiment, between -1 and +1, where -1 is the lowest possible value, and +1 is the highest possible value.

Generating the Graph

This graph was rendered using Google Charts and then dynamically modified with a script written in D3.js. Because this graph is dynamic, it is possible for the user to enter any search term, and the system will fetch relevant Tweets, then run the sentiment scripts on those Tweets and generate a similar looking graph. This process is fully automated, and typically takes 5 – 8 seconds.

Results

There are several findings from this research.

Firstly, there is a clear relationship between the length of the input string (in this case a Tweet) and the accuracy of the results. The longer the input text, the closer together the NLU, dictionary and human results, in most cases. This is because a better understanding of what it being conveyed can be grasped in longer sentences.

The dictionary-based results tended to produce more neutral values, whereas the NLU method was able to distinguish positivity and negativity in most tweets. This is because it is able to interpret actual semantic meaning from the sentence as opposed to just looking and the positivity of words.

The overall average sentiment produced by the NLU dataset for the Edward Snowden dataset was 0.028 (very close to neutral as a lot of very positive and very negative tweets cancelled each other

out), and the average for the dictionary based approach was 0.019 (again very neutral). This difference of 0.009 show's that the two methods, despite being very different overall produced results not that far out.

Finally, there were some cases (on other datasets), where the sentence was using very positive sounding words to convey a sarcastic message. In some cases, the NLU method was able to distinguish this, and gave an appropriate sentiment score, however, the dictionary-based approach failed miserably.

2.2.6 Summary of comparison between different SA approaches

The following radar chart illustrates how dictionary-based method compares with the NLU approach. The data was calculated based on the custom written dictionary approach mentioned above, and the HP Haven NLU sentiment analysis engine.

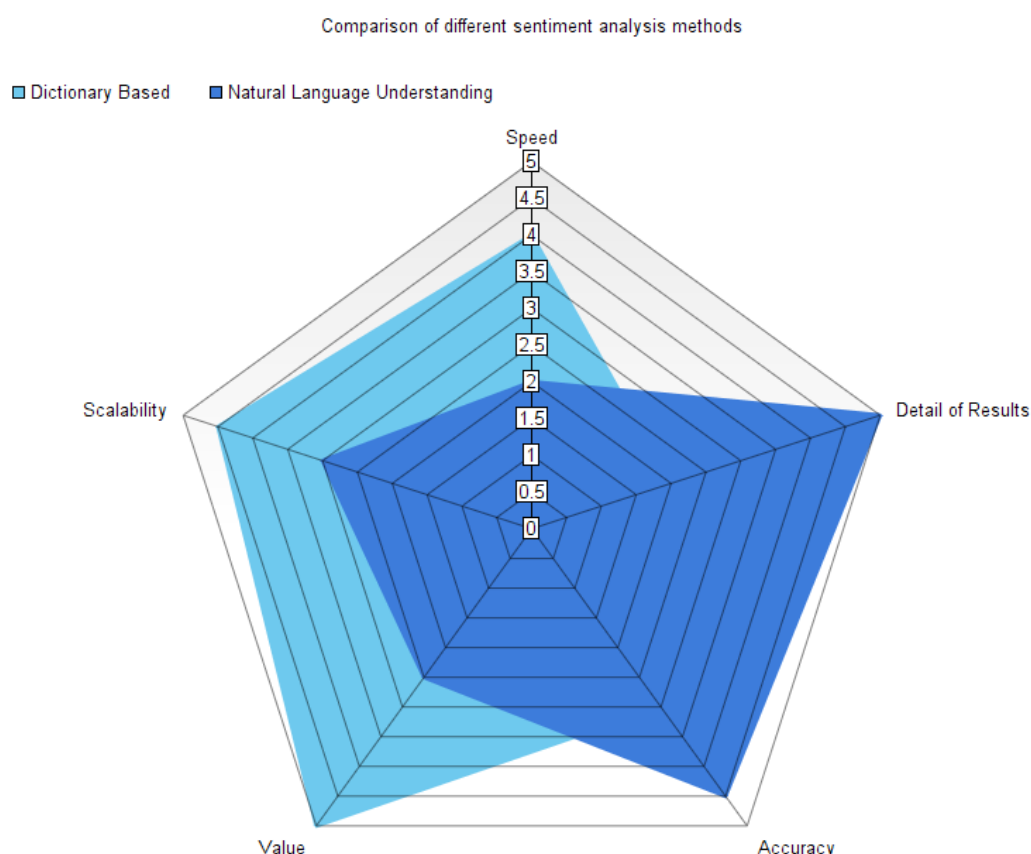


Fig 3 Radar chart showing conclusion of different SA methods

The radar chart illustrates how although NLU SA is significantly more accurate and returns very detailed results, it is certainly not scalable for larger solutions, nor is it fast (hence not suitable for real-time data), and is not cheap to implement and maintain either.

In conclusion, although the natural language understanding approach is able to deliver more accurate results and distinguish a wider variety of emotions, it takes a lot longer to complete each request, and also requires considerably more computing power, both of which means that it is less cost effective and scalable for larger solutions.

3 METHODOLOGY

The methodology section is split into five parts; development plan, development tools, code documentation, testing, front-end and back-end.

3.1 DEVELOPMENT PLAN

3.1.1 System Development Life Cycle Process

The system development life cycle is the process of splitting of the software development into distinct phases, to better the planning and management of the overall project. Following a system development life process strictly greatly increases the likelihood that all the intended deliverables will be completed on time, and meeting requirements.

The methodology used in this project followed the principles of agile, more specifically personal-SCRUM (one-man agile). This is an iterative approach, where the project will be divided into a set of phases, called sprints. Each sprint had a set of requirements presented in the form of user stories and acceptance criteria. The sprint was only marked as complete once each story has been developed, implemented and tested (or descoped). User stories were prioritised and given a complexity estimate before each sprint, to ensure the best use of time and resources.

Several other options were considered, before agile was chosen. Another common approach is waterfall. Waterfall is much more rigid and better used when all requirements are known beforehand, and a single phase of planning, development, implementation and testing takes place. It can be easier to predict and plan for, however considerably less adaptable. Since a key component of this project is research and user experience testing, flexibility is key. The final deliverable may have slight differences from the proposed deliverable, but will still meet all requirements, and will probably be better than the original plan.

The development phase of the project comprised of twelve, two-week sprints. During each sprint, the code was tested against each of the acceptance criteria specified for the user stories. The iterative fashion of this approach ensured that the most important requirements were prioritised, and this in turn made it possible to finish the project on time.

Personal-SCRUM is a type of agile development used in one-person teams. Ravikant explains how agile can be applied to a single person team in the paper: Extreme programming for a single person team. He explains how it is based around simplicity, communication, feedback, and courage. There are several key parts of agile that it is essential to stick to while working in a one-man team.

- Test Driven Development (TDD)
- Refactoring
- Continuous Integration
- Doing the simplest possible to make the solution work, then refactoring
- Automated deployment

3.1.2 Risk Analysis

A risk is categorised as an unknown that cannot be predated or prevented but whose likelihood may be estimated. This section identifies the potential risks that could have had an effect on either development or the final solution, and the plans that were taken to minimise their impact on the project. Identifying risks before development allowed for better control over the project hence made it easier to manage uncertain events.

The following is based on Boehm's (1991) categorisation of software risks, and puts into practice some of the principles from the ISO 31000:2009. Boehm describes how important risk management is to avoid project disasters, rework and overkill.

Risk Identification

The following risks were identified relating both specifically to this project and more broadly from Boehm's top ten project risk factors.

- Change in Twitter API T&C's – The whole project was dependent on Twitter data. Twitter have very strict and comprehensive T&C's which do change quite regularly in accordance with law.
- Reaching data limits (Google Places, Twitter, HP HavenOnDemand) – In order for the system to use live or real-time data, API's will be required, and they all have data limits which cannot be exceeded without paying a fee.
- Running out of budgeted computing resources – cloud computing was used, for this there is a limited amount of computing resources that can be used to stay within budget.
- Lack of specialist skills – the project used a lot of brand-new and cutting edge technologies some of which have limited information available on the net which could have posed as a problem, while upskilling for development, and for future maintenance.
- Change in UK law and regulations relating to data privacy – a lot of user data from social media streams is being utilised in the system. This data being made public requires law not to change drastically over the time frame of the coming few years.
- Management of bugs and unexpected glitches – as with any project there are bound to be some unexpected bugs, which can take a variable amount of time to fix.
- Poor time estimations – the project schedule put forward in the proposal was ambitious, and there potentially could have been problems sticking to it in the face of unexpected events.

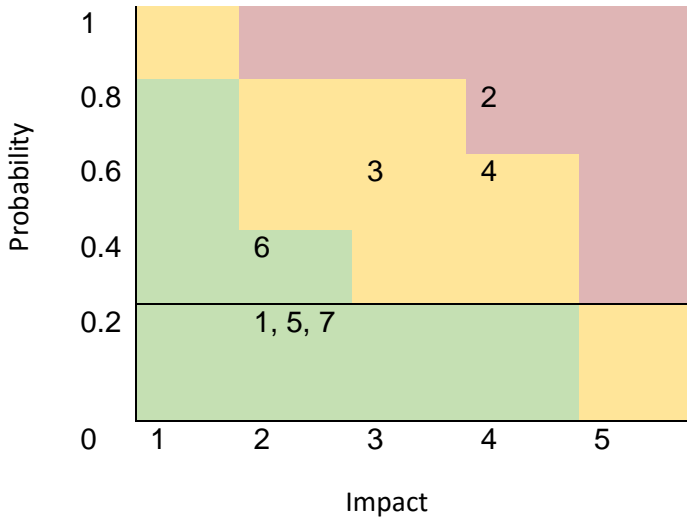
Risk Assessment

Risks were calculated using Bernstein's (2005) method. Risk exposure = likelihood * impact. Where the likelihood is between 0 and 1, and the impact between 1 and 5. Quantifying the risks allowed the project risks to be prioritised.

Risk	Likelihood	Impact	Risk Exposure
1. Change in Twitter API T&C's	0.2	2.5	0.5
2. Reaching data limits	0.8	4.5	3.6
3. Running out of budgeted computing resources	0.5	3.5	1.75
4. Lack of specialist skills	0.6	4	2.4
5. Change in UK law and regulations	0.2	2	0.4
6. Management of bugs	0.4	2	0.8
7. Poor time estimations	0.3	2	0.6

Table 1 Table showing the likelihood, impact and risk exposure of each identified risk

Risk Severity Matrix



The risk severity matrix highlights which risks should be prioritised based on their likelihood to occur and the potential impact they could have had on development or the final project.

Table 2 Table showing probability against impact of each risk

Risk Response

The risk assessment highlighted three risks which attention should be paid to.

1. Reaching data limits on the utilised API's
2. Running out of computing resources
3. Lack of specialist skills

Reaching the data limits for Google places, Twitter and any other API's utilised could have had a major impact on the final project. A risk reduction approach was taken, where possible data was cached to minimise requesting similar data. Data requests were only sent when necessary, and background tasks are stopped when the application is idle. In the future it may be necessary to limit the amount of requests from each user, to avoid a single user sending off 1000's of requests to drain resources.

The risk for running out of computing resources can be transferred if necessary. A cloud computing server with much higher specs can be utilised, or a plan with unlimited bandwidth can be selected. At present that hasn't been necessary, and nor will it be in the short-run.

The lack of specialist skills was a risk that just had to be accepted. I upskilled as much as possible and used available resources to gain the knowledge needed.

The majority of the other lower priority risks were reduced by allowing for contingency time. The amount of contingency allowed for each stage has been highlighted in the GANNT chart. Approximately 10% or one week extra for each separate phase of the project.

3.1.3 Project Schedule

Development was started on the Saturday 20th of June 2015. The following table gives a brief outline of tasks carried out. Each section was followed by a 10% contingency time. The three development sections (e), (f) and (g) were broken down further into sprints. About 5 – 8 hours work was done per day (30-40 hrs a week) during development.

Work Break-down

ID	Duration	Section	Tasks
(a)	4 weeks	Investigation and Research	Investigation of the Problem
			Investigation of Previous Approaches
			Literature Review
			Research different sentiment analysis techniques
			Research into development technologies
			Research into server side technologies
(b)	1 week	Write Project Proposal	
(c)	3 weeks	Project setup and configuration	Determine and create file structure
			Create git repo
			Set up test environment and CI testing
			Set up automated code reviews and dependency checking
(d)	1 week	Write interim report	-
(e)	6 weeks	Develop backend modules	Following the agile process with TDD and publishing for each sentiment-analysis
			fetch-tweets
			stream-tweets
			remove-words
			place-lookup
			hp-haven-sentiment-analysis
			tweet-location
			find-region-from-location
(f)	2 weeks	Develop Express app	Create the main application from above node modules
(g)	6 weeks	Develop front end	Page structure
			Heat map
			Region map
			Raw tweets
			Word cloud and scatter plot
			Comparison
			Trending
			Timeline
			3D globe
			Entity extraction
			Tone identification
			Search
Homepage			
(i)	2 weeks	Publish	Put all final components together
			Ensure all documentation is up-to-date
			Upload and publish to public web server
(j)	4 weeks	Write final report	-
(k)	2 weeks	Contingency	Ensure server application is running smoothly
			Fix any reported bugs or issues

Table 3 Table showing a breakdown of all work to be completed in each timeframe

Gantt Chart

The chart below shows the time schedule that the project followed. Dark orange represents planned time, where the contingency was sometimes used when ahead or behind schedule.

Gantt Chart

■ Duration ■ Contingency

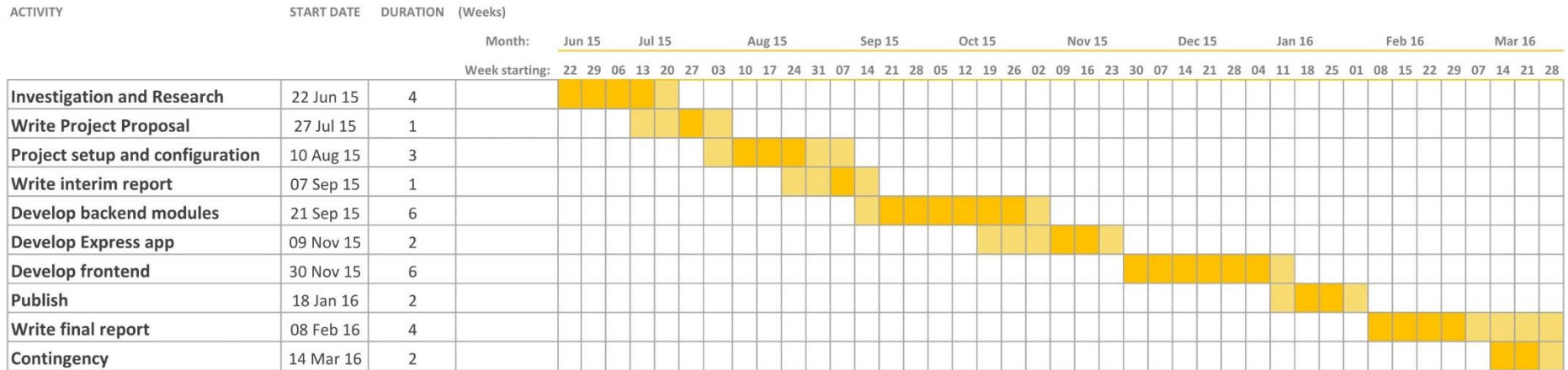


Table 4 Gantt Chart showing time allocated to each development task

Sprints

Sprint 0 – Project setup

Sprint 1 – Create base application and empty database

Sprint 2 – All tweet handling backend modules

Sprint 3 – All location and utility backend modules

Sprint 4 – Sentiment Analysis module

Sprint 5 – Integrate backend modules into application

Sprint 6 – Link blank frontend to backend to make twitter sentiment results

Sprint 7 – All geographically based sentiment data visualisations

Sprint 8 – All keyword based sentiment data visualisations

Sprint 9 – Timeline, trending and comparison data visualisations

Sprint 10 – Implement real-time functionality into data visualisations and create txt page

Sprint 11 – Home page and search page

Sprint 12 – Final testing, code quality, UX review – then publish to public cloud server



Table 5 Table showing which user stories will be completed in each sprint

The final sprint will run from 15th of February into mid-March 2016, and will cover all not yet complete user stories, as well as UX testing and further refactoring and finalising documentation.

3.2 DEVELOPMENT TOOLS

3.2.1 Developing

Version Control

Version control is a system that records the changes made to a file or set of files over time, allowing the developers to recall specific versions later (Linus Torvalds 2016). Large projects can get complicated, and it is very useful to be able to track changes, and create branches and development versions of the current code base. As the name suggests it allows the developer to have 'versions' of their code, which can make backtracking possible if necessary.

Git is a distributed version control system; it works by taking snapshots of the changes in files, meaning if something is unchanged, the previous version can be referenced. This makes it incredibly lightweight and fast. Several VC systems were considered including SVN, CVS, TFS and Mercurial. But for the reasons listed above Git was selected for this project.

Git can be run either on a server or locally. GitHub is a remote server running an instance of Git. It also acts as a community for developers and has a public GUI. GitHub is the most commonly used platform for developing and publishing open source code. Since one of the aims of this project was to develop and publish a series of cutting edge open source modules, GitHub was an ideal choice. All open source code from this project is therefore hosted on GitHub.

IDE

An integrated development environment (or IDE) is an application that facilitates the fast writing of quality code, through a set of features beyond those of a plain text editor. Several tools were considered including a variety of cloud-based and local software packages. JetBrains's Web Storm (the equivalent of IntelliJ, but for the web) was selected. It has a range of features that will aid the developer in writing good quality code. Including inbuilt debugger, local server integration, intelligent autocomplete and syntax highlighting, VCS features and a very customizable UI. It was important to select a good IDE here, as about 1000 hours was spent developing.

Development Server

A local offline development server was utilised for fast development, without the need to push to a remote server to preview. It was then published on a CentOS VPS (see more in implementation section)

3.2.2 Building

A requirement of the proposed deliverable is that the application must be efficient, as it handles a large amount of data and graphics. In order to achieve this the source code written by the developer will have to go through a process to streamline and compile it into its lightest form. A number of tasks have to be run on the original code base to get it in its deployable form. A task runner will be configured to automatically execute these tasks during development. This section briefly outlines how the task runner and build tools utilised were configured, and why they were selected.

The build tool

A build tool is a utility program to automate the creation of an executable application from source code. For node applications, there are several large players in the market; Grunt, Gulp, Brunch and Broccoli. Before starting the project these build tool were compared on ease of configuring, community support, available plugins, setup speed and most importantly speed and efficiency of building. The final choice for this project was Gulp.

The primary reason for selecting Gulp is that it is a streaming build system, which means that unlike the others it deals with streams as opposed to creating temporary files – that, of course, can have a major improvement on speed. Gulp also has a very large community and many plugins available to cover a range of build tasks.

Compiling Server-side Scripts

The server side code is Node.js. This was written in CoffeeScript, so one task of the build system was to watch, check and compile CoffeeScript into efficient JavaScript. The build system was configured to carry out a number of other tasks regarding the server side scripts, which are outlined in the acceptance criteria of user story TSV-T075.

Compiling Client-side Scripts

The client-side scripts were also written in CoffeeScript. For maximum code reuse, everything was written in a very modular fashion and object-oriented where possible. For this reason, it was necessary to bundle client-side scripts for each page up together into a set of single minified files. Browserify was utilised to achieve this. This also made it possible to use external JS libraries on the client-side and share client-side and server-side code. The Browserify configuration also covered all the acceptance criteria listed for TSV-T080

Compiling Styles

All styles were written in Less and SASS. The build process then checked, streamlined and compiled this into minified CSS. It also removed obsolete styles (that were not specified in the HTML, Jade, JS), and if there were any lint results they were logged. This related to user story TSV-T076.

Images

As specified in TSV-T078, all images were streamlined and piped to the public folder. It is important that graphics are no larger than they need to be, as they can greatly slow down load speeds.

Test Running

As TSV-T084 outlined, the build environment also runs the test scripts when that particular file changes. More on this is specified in the test strategy.

Cleaning Workspace

All obsolete files are removed following the acceptance criteria in TSV-T079.

Watch

The build setup is configured to build every time a file changes. It only runs the tasks relevant to that file, so is very fast. This follows the acceptance criteria in story TSV-T082.

Logging

After a build, if there are any code errors, efficiency suggestions or quality guidelines not met, then it shall be printed the run log, or on the console. During development this ensured the developer was made aware immediately if something was not perfect. The size of each bundle is also outputted, as this needs to be monitored for efficiency. This is in accordance with story TSV-T081.

Syncing Browsers and Automatic Server Restarts

For ease of developing across many browsers, screen sizes and platforms all instances of the application during development were kept in sync. When the developer scrolls down in Internet Explorer, it should also scroll on Chrome, the Android device and any instances in real-time. The same should apply if the developer clicks a button, all interactions should be fully synced live across many devices during development. When a change the backend is made, the server should automatically restart. After restart, all browsers should automatically refresh. This follows user story TSV-T083.

Provide an easy interface for running the above tasks

The developer can run any of the following commands during development:

Specific Tasks

- `gulp generate-config` - before first-time running of the project, run this command to generate configuration files for API keys and environmental variables
- `gulp scripts` - compiles all scripts
- `gulp browserify` - generates all server side Browserify bundles
- `gulp styles` - compiles all styles into CSS
- `gulp images` - streamlines and pipes all images to the public directory
- `gulp watch` - watches all development files and runs appropriate tasks when they change
- `gulp clean` - wipes the output directory and removes obsolete files

General Tasks

- `gulp build` - This builds the project fully, this includes cleaning the working directory and then all tasks that must happen for CoffeeScript, JavaScript, CSS, images, HTML and Browserify tasks.
- `gulp nodemon` - Runs the application on development server on the default port
- `gulp test` - This runs all unit and coverage tests, printing a summary of the results to the console and generating more detailed reports into the reports directory.
- `gulp` - this is the default task, it checks the project is configured correctly, build ALL the files, run the server, watch for changes, recompile relevant files and reload browsers on change, and keep all browsers in sync, when a test condition changes it will also re-run tests - a lot going on!

3.3 TESTING

A series of open source node modules were developed and published as part of the intended deliverables. A requirement for these was that they must be working, stable and efficient. This was achieved through testing. Following a test strategy is essential for producing high quality code, for this reason a lot of planning went into the test plan and environment.

3.3.1 Test Driven Development

Test driven development (or TDD) is a process where the tests are written before the code is developed. The test cases are written based on the user stories, and most the logic on the application is developed in the tests, so when it comes to writing the code it should be a very quick process.

TDD was chosen as the testing methodology, as it has several key advantages. Firstly, it will ensure the code that is written is structured, as the structure must be determined before the code can be written. Secondly it helps the code fit with the user stories, since the tests will be based from the user stories. It also creates a detailed specification. Most importantly less time is spent debugging and fixing bugs, as code is written to pass tests.

3.3.2 Unit Testing

Unit testing involves writing a series of very thorough tests to cover each module, function or method independently as a unit. Each function should be checked that it produces the expected output with a variety of hardcoded inputs. This included testing error handling, and borderline and unexpected inputs. After the unit tests were written the UT process was automated, so every time a method is changed, the tests are rerun to check that it still produces the correct output with various inputs.

Unit testing has a lot of benefits to software development, firstly any potential bugs or failures will be identified before that function gets integrated with the larger application. Developers can verify their code still works as expected as they refactor and change parts, in the same way, unit testing can prevent future changes from breaking functionality. It also helps the developer understand the code, and gives instant feedback when something is not working as it should be. It was for these reasons that unit testing was thoroughly implemented in the modules for this project.

3.3.3 Behaviour Driven Development

This is the same principle as TDD, but involves writing tests with a more functional point of view. The syntax used to write the tests tends to be more like English, and the tests follow very closely to the user stories. Following BDD ensured that the code was written followed the user stories and acceptance criteria.

3.3.4 Pass / Fail Criteria

Before each sprint could be marked as complete, it had to have met the following test criteria.

Test Type	Pass Condition
Functional Testing	All acceptance criteria must be met, checked and documented
Unit Tests	All acceptance criteria must be met, checked and documented
Integration Tests	100% pass rate after every commit
Coverage Tests	80% or greater
Code Reviews	B grade/ Level 4 or higher. Ideally A grade/ Level 5 if possible.
Dependency Checks	Mostly up-to-date dependencies except in justified circumstances.

Table 6 Table showing the pass fail criteria for each sprint

3.3.5 Documenting Results

A status of all unit tests, coverage tests, dependency checks and code review will be displayed in the form of badges on the repository readme. Each will be linked with the appropriate service, so will update live. This will indicate immediately as soon as a test is failing or a dependency becomes outdated, and will be very useful for other developers.

Detailed test reports for each testing method are generated and saved to the reports directory. These reports have been configured to show very detailed test results that were analysed towards the end of each sprint. A breakdown of these results can be found in the appendix.

During development, as a piece of code is changed, the relevant unit tests are run and the results are outputted onto the console. This gives the developer instant feedback of the stability of the application in its current state.

3.3.6 Automated Continuous Integration Testing

Integration testing is the phase of development where the individual modules are combined and tested as a whole. The purpose of this, is to verify the program still works reliably and as it should as a whole. For this project every module is unit tested, and so automated continuous integration testing was utilised to ensure everything continued to work as a whole during development.

Travis-CI is a free open source continuous integration service that builds and tests software packages that are hosted on GitHub. A configuration file was written in YAML and this specified how the environment should be set up. Travis also has the benefit of running the project across multiple different server setups, to ensure it will work within any environment.

3.3.7 Coverage Testing

Coverage testing is the process of determining what proportion of the source code is covered by the unit tests. Coverage tests can ensure that limited code is missed out from the test plan, in turn increasing the stability of the application. For this project the threshold was set to 80%, so each module must have at least 80% coverage in order to meet requirements. Istanbul is a utility program which analyses unit tests and source code to produce a detailed coverage report. It is open source and has been integrated into the build configuration to run whenever tests are updated.

3.3.8 Dependency Checking

Since there are quite a few external dependencies that came together to make each component of the project possible, it was important to ensure that all the current dependencies are stable and have no bugs that could affect the running of the service. For this automated dependency checking was implemented with David-DM. When a dependency was no longer in date a notification was triggered.

3.3.9 Automated Code Reviews

Code reviews can pick up on bad practices and inefficient code, such as including dependencies and not using them, variables in the wrong scope, poor identifiers, not following convention etc. All of this information can help write better quality code. Code Climate is an open source utility that checks complexity, duplication, security, style and many other important aspects to ensure code follows a specified style guide. This tool was utilised to monitor the quality throughout the duration of the projects development.

3.3.10 Assertions

An assertion is an expression which encapsulates some testable logic. An assertion library, in node is a software package which provides functionality such as quickly testing if a condition is true, or if a value is in an array, or if length is greater than a value and many more features. As part of the unit testing requirement, assertions must be made (where possible) to express each user stories. Several options were considered, but the Chai TDD/BDD library was selected for this project, as it is very lightweight and flexible.

3.3.11 Test Framework

A test framework provides a structure for which the tests can be based on, and run. Several options were considered for this project, and Mocha was chosen. It is a feature-rich and well established JavaScript testing framework. And required in order to store, write and run the tests in a structured way. Using a framework will also make using various testing plugins easier to use neatly.

3.3.12 Stubs, Spies and Mocking

For the tests, it would not have been good practice to have any network calls, so Sinon.js was used to stub the data that would have been returned for each network call. Spies will also be used to test the functionality of methods.

3.3.13 Headless Testing

For running functional frontend tests without having to use a browser, it was automated in the same way as the other tests and can test the integration with other frontend libraries. PhantomJS was used for this, as it also provides network monitoring utilities that can help cut down page load times.

3.3.14 Testing HTTP services

SuperTest is an agent driven library for testing node.js HTTP servers using a fluent API. It was used for testing HTTP servers and checking the routing for the Express web service, to ensure with a given URL and parameters returns the expected output.

3.4 CODE STYLE AND IN-CODE DOCUMENTATION

3.4.1 Documentation

Good documentation is essential to ensure other developers can read, understand and modify the code.

All code has in-code documentation in the form of comments where appropriate.

For every module that was produced and published publicly, a detailed readme was written up and packaged. It outlined how to use the module, an API description of that module, how to run the tests and how to develop or modify for the module.

Unit tests can also be used as a form of documentation, and so detailed test cases were written for each published module.

If code is written following good style, then the code itself can act as a form of documentation. Therefore, many measures were taken to ensure the code was the best possible quality. These are outlined in the next section.

3.4.2 Code Style

The majority of the logic for this application was written in CoffeeScript. It is important that all code follows best practices, but more important that it remains consistent throughout the project.

For this reason, a style guide was followed very strictly. Furthermore, an automated task (coffee-lint) was incorporated into the gulp build setup, to automatically check (where possible) that code does not break any of the specified rules.

A summary of the style guide can be found in the appendix.

3.5 FRONT-END PLAN

This section will briefly outline the screens that will be included in the application, and link to which user stories and acceptance criteria apply to each.

3.5.1 Data source for each page

All pages (with the exception of that start screen) will follow the same format in terms of data source. Upon landing on the page, results will be calculated on tweets that are cached from the backend streaming engine, and up to 60 minutes' old. The user can then make a search (or add `/[search-query]` onto the URL). This will fetch a fresh set of tweets, and render the data visualisation based on this new data. This is explained more in the backend section of the methodology.

3.5.2 Frontend Technologies

Google Maps

The sentiment heat map, will make use of the Google Maps and Google Places API.

Socket.io

Socket.IO enables real-time bidirectional event-based communication, it will take care of aspects like web sockets, and ensure maximum browser support. Socket.io will be implemented in all data visualizations and charts that require live data to be displayed and updated in real-time. Socket.io will signal to all subscribed listeners, whenever there is a change in the backend data. The frontend subscribers will then be able to calculate what changed, and re-render the appropriate component.

It is a good choice since it handles graceful degradation to numerous technical alternatives for real-time functionality, and it also handles browser inconsistencies.

Materialize

Materialize is a modern responsive frontend framework, which implements many of the design standards of material design (released by Google at IO/14)

D3

The majority of the charts and data visualizations will be coded in D3.js (but written in CoffeeScript). D3 (or Data Driven Documents) is an advanced JavaScript library for manipulating documents based on data. It allows for web elements (such as SVG and HTML) to be bound to data. (See more at <https://d3js.org/>).

D3 was chosen, because it is focused on binding DOM elements to the data, and is a bare-bones language, meaning the highly customizable documents can be created. Furthermore D3.js is written in JavaScript, and uses a functional style meaning code reuse is seamless. For this project CoffeeScript will be used as opposed to JavaScript, but the same applies.

3.5.3 The Home Screen

This will be the initial landing page for the application.

As the wireframes show, it was split into two parts, and there was a parallax scrolling animation between the two. The upper section of the home screen will give a brief introduction to the application and provide an input field for the user to enter a search term. The lower section of the screen has a link to each of the data visualisations across the site.

Aims

- To give a brief overview of what the application is, does and can be used for
- To provide navigation to every other page on the site
- To load quickly and present interesting information to the user to decrease bounce rate

User Stores

The home screen must meet the acceptance criteria laid out in TSV-H036, TSV-H037 and TSV-H038 (see appendix for user stories)

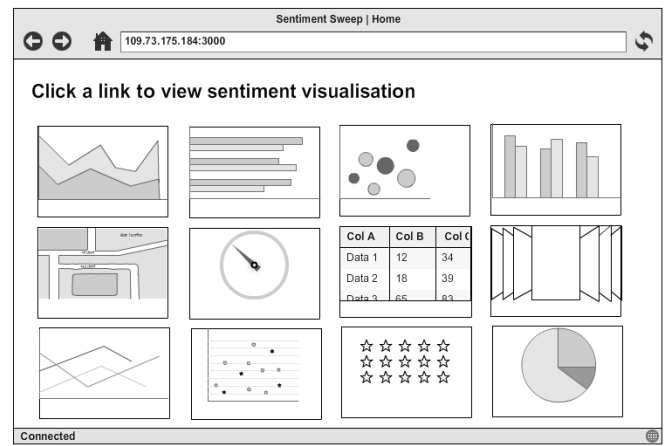
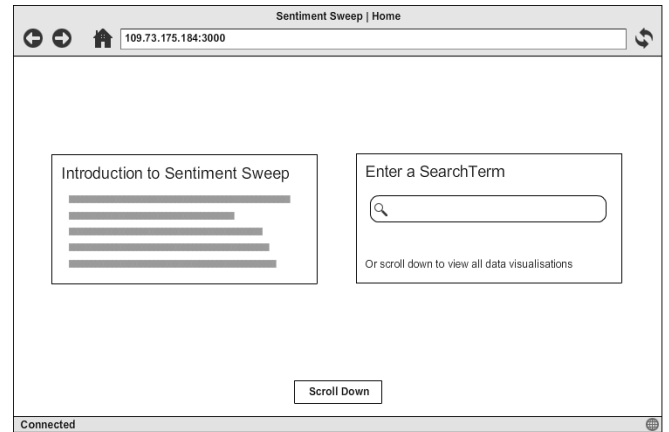


Fig 4 Wireframe for home screen

3.5.4 The Heat Map Screen

A real-time and interactive geographical map, with a dynamic heat map overlay varying in colour to indicate which areas are more positive or negative

Aims

- To give an overview of which parts of the world, a country or a city are more or less positive about a certain topic, or overall
- To allow the user to zoom in, and drill down to analyses a more specific area and read actual tweets
- To illustrate real-time regional events, either positive or negative

User Stories

TSV-C009, TSV-C010, TSV-C011, TSV-C012, TSV-C013, TSV-C014.

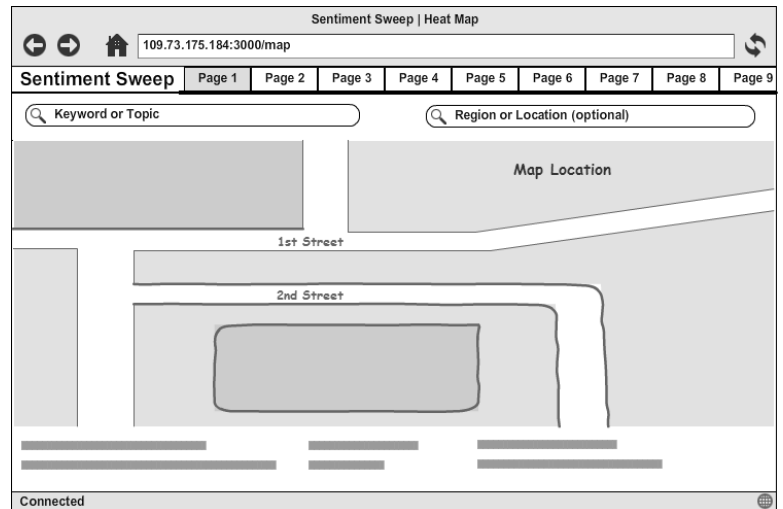


Fig 5 Wireframe for the heat map screen

3.5.5 The Regional Sentiment Map

Very similar to the heat map (above), but show's overall sentiment for each region. This can be very useful for getting a quick overview, rather than analysing every heat patch

Aims

- Provide a geographical snapshot of sentiment towards a given keyword
- Simple to look at
- List most and least positive regions
- Links for drilling down to gain more details, but not actually displaying details on this screen

User Stories

TSV-I039, TSV-I040, TSV-I041

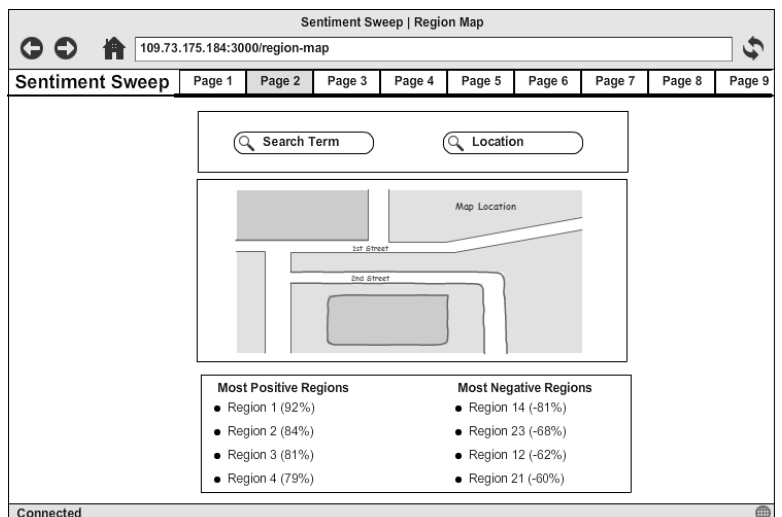


Fig 6 Wireframe for the region map screen

3.5.6 Time Line

An interactive area chart displaying average positive and negative sentiment (y-axis) against time of day (x-axis) either overall or towards a topic

Aim

To show what times of day people are most positive or negative

User Stories

TSV-O055, TSV-O056

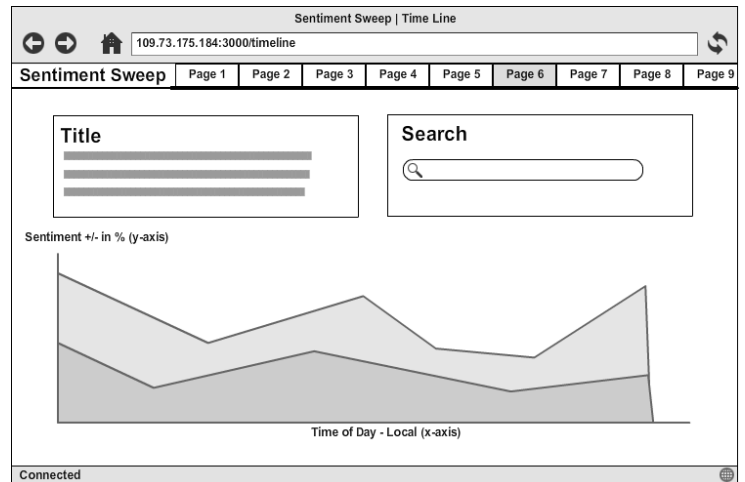


Fig 7 Wireframe for the timeline screen

3.5.7 Raw Tweets

Often after viewing the various data visualizations around a topic, a user may want to just view the original plain-text tweets, so they can read exactly what people are saying. Each tweet will be displayed in real-time, with location, time and sentiment. Keywords will be in bold to allow the user to scan read quickly

Aims

- To allow the user to view the text of tweets, as opposed to a data visualisation
- To allow for users to monitor in real-time what people are saying about their chosen topic
- To make it possible for the user to skim read relevant twitter data

User Stories

TSV-H046, TSV-H047, TSV-H048

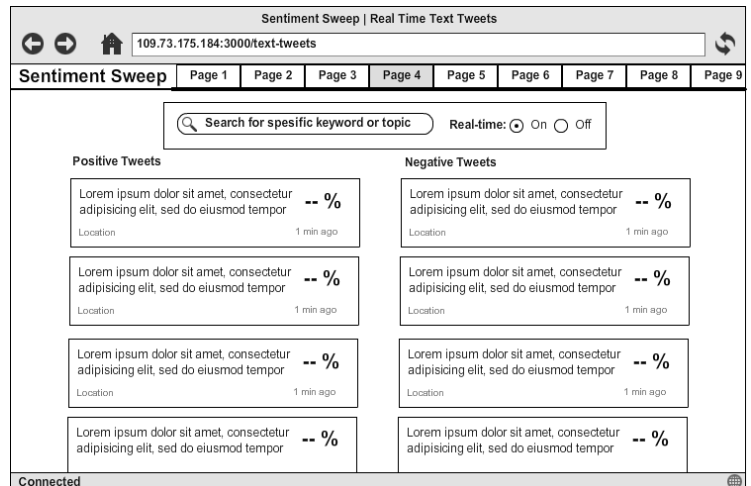


Fig 8 Wireframe for the raw tweets screen

3.5.8 3D Globe

A real-time 3D globe showing both sentiment and volume for geographical nations across the planet, and updating live.

Aim

To visually illustrate how both sentiments with magnitude, and volume of data varies across the planet

User Stories

TSV-N052, TSV-N053, TSV-N054

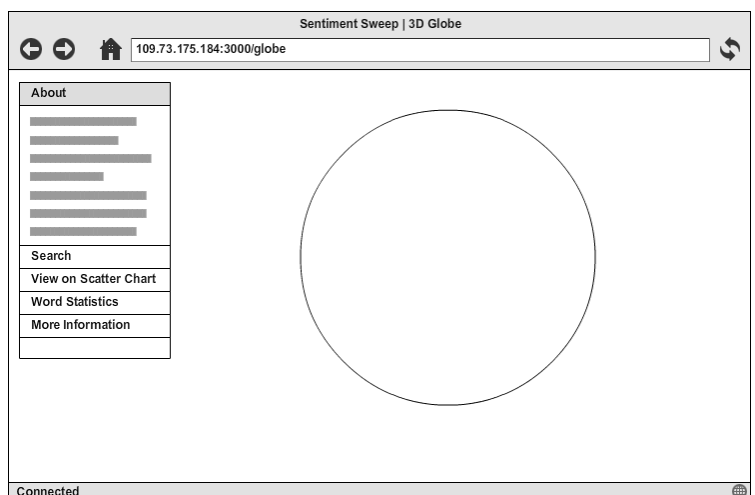


Fig 9 Wireframe for the globe screen

3.5.9 Word Cloud

The word cloud will be a collection of the most commonly used words in a set of tweets. The size of the word will represent volume of use, and the colour will show average sentiment of the tweet the word came from

Aims

The aim of the word cloud is to illustrate which key words with a strong sentiment value are commonly used in a set of tweets, relating to a topic. This will enable the user to better understand why the overall sentiment is what it is.

User Stories

TSV-L049, TSV-L050

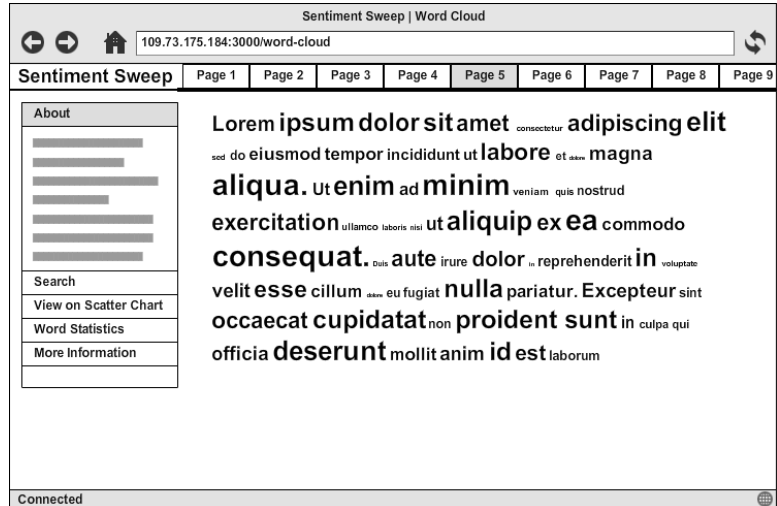


Fig 10 Wireframe for the word cloud screen

3.5.10 Word Scatter Chart

Represents similar data to the word cloud, but in the form of a scatter plot, which can allow for much higher volumes of data, and can be more practical for inferring trends. Sentiment will be on the y-axis, while frequency of use on the x-axis. The user can hover over a point to view associated word.

Aim

To visually illustrate trends between commonly used words and sentiment of tweets in a dataset

User Stories

TSV-M050, TSV-M051

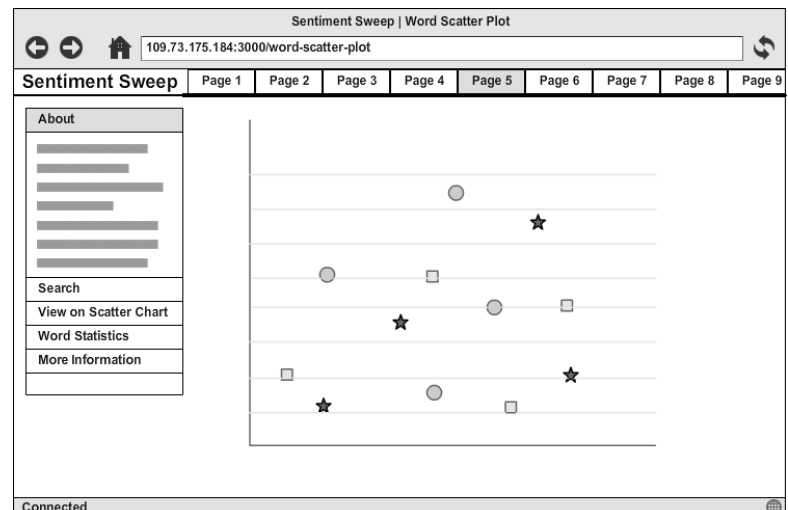


Fig 11 Wireframe for the word scatter plot screen

3.5.11 Trending

This screen shows the keywords, hashtags and users that are currently trending on Twitter at that time. It also shows the volume of tweets for each trend, and calculates and displays the overall sentiment.

Aims

- To provide the user with the most up-to-date topical sentiment information from their local area
- To show an overview of what's currently trending both worldwide and at a custom location
- To show which topics are positive and which are negative, as well as volume
- To provide links to the search screen for each trend

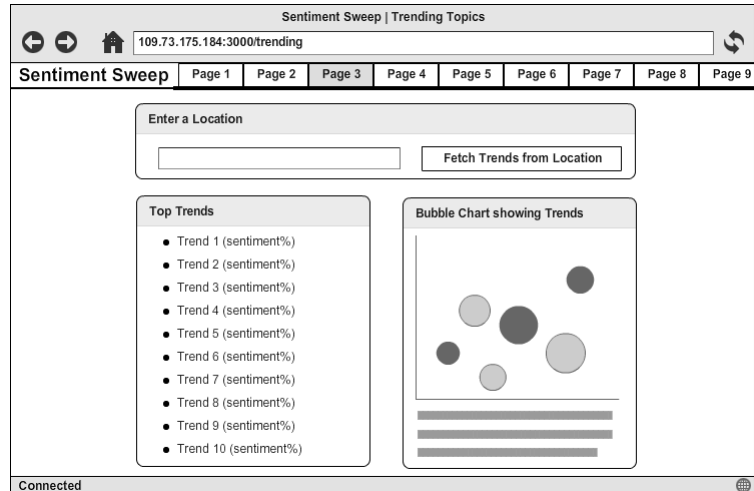


Fig 12 Wireframe for the trending screen

User Stories

TSV-J042, TSV-J043, TSV-J044, TSV-J045

3.5.12 Comparison

Often users will want to benchmark the results from their query against other similar topics, or competing brands. The comparison page allows up to four topics to be compared on a single screen.

Aims

- To allow the user to compare similar topics in one place
- To get a snapshot of how each topic is currently doing
- To provide links for further details for each topic

User Stories

TSV-P057, TSV-P058, TSV-P059, TSV-P060

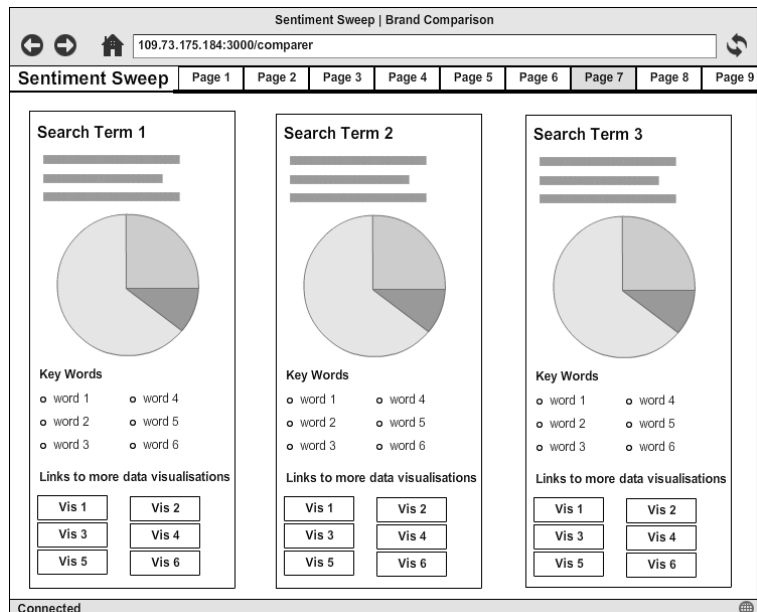


Fig 13 Wireframe for the comparison screen

3.5.13 Entity Extraction

The entity extraction screen shows the results of the key label sequences recognised in a set of tweets about a certain topic. Results will be summarised in the form of a dynamic Sankey diagram. Below that will be more detailed textual results. Entities can be anything fitting into the defined categories e.g. places, people, languages, objects, companies, films...

Aims

- To clearly show which entities are commonly referred to when people are talking about a topic
- To provide visual representation of the proportion of tweets talking about each entity

User Stories

TSV-R065, TSV-R066

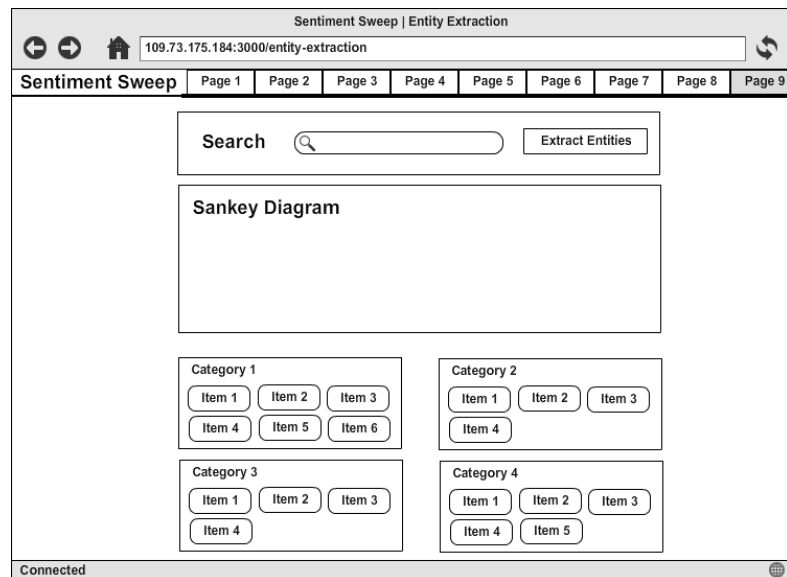


Fig 14 Wireframe for the entity extraction screen

3.5.14 Tone Identification

This screen will visually present the results from the tones of speech conveyed in a set of tweets.

Aim

To visually display the tones of speech from a set of tweets

User Stories

TSV-Q063, TSV-Q064

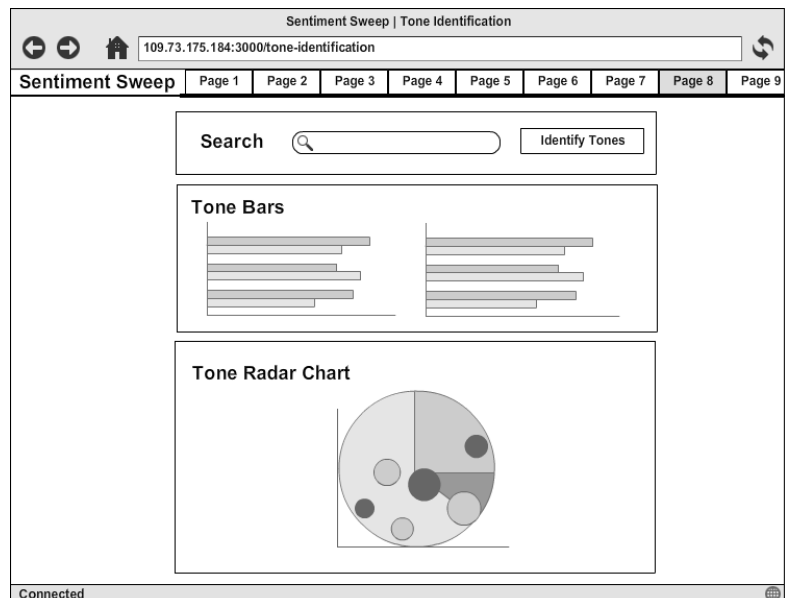


Fig 15 Wireframe for the tone identification screen

3.5.15 The Search Results Screen

Displays the results for a search made from the home page.

Aims

- To provide an overview and summary of the different sections of the application on one page
- To act as a starting point, with links to other parts of the application customized specifically to the topic searched for by the user

User Stories

TSV-S067, TSV-S068, TSV-S069, TSV-S070, TSV-S071, TSV-S072, TSV-S073, TSV-S074

Short Description of Wireframe

The wireframe on the right shows each of the eight sections. The first is a summary of the sentiment results, including a gauge showing average sentiment, and a list of most commonly used keywords and their sentiment.

The second section is a summary of the tones of speech identified in the set of tweets, with a link to show more detailed results.

Similarly, the third section is a summary of the key entities extracted from the set of tweets, each entity will be clickable, and there will be a see more link.

The fourth section will show how the current topic compares to the averages on the rest of twitter, with the option to compare the current topic with up to three other topics.

Proceeding that is a small set of the raw tweets with some additional information, and a load more tweets button to show more.

Finally, there is a customized link to each other part of the site, specific to that particular topic, followed by an input field to make a new search.

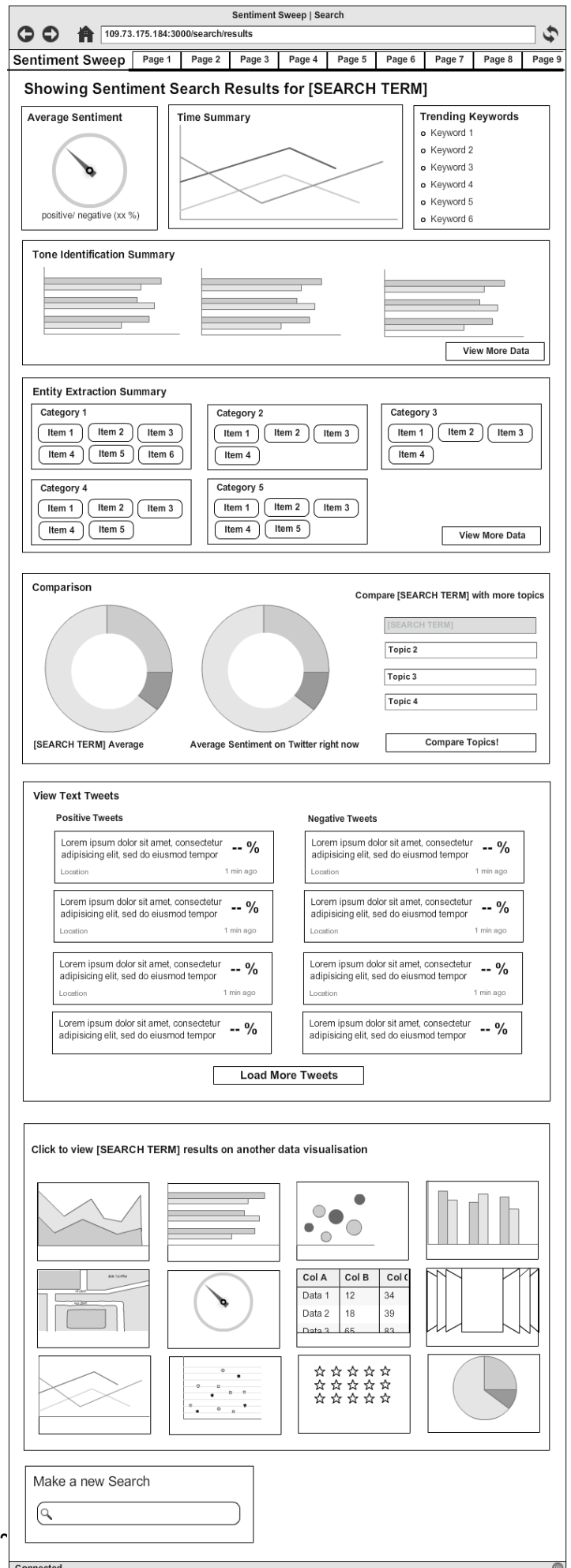


Fig 16 Wireframe for the search page

Backend Plan

This section briefly outlines how the backend was developed, and how it integrates with the project as a whole.

3.5.16 Data Cache

As specified in the proposed deliverables, the application must be capable of showing cached tweets from the past 60 or so minutes. This will enable a data visualisation to be loaded from local data on initial request, and then pipe new data to it after load. The reason this is required, is it will drastically decrease page load times, and increase usability of data.

Several database solutions were considered during project planning, but the decision to use MongoDB was made due to its scalability and document-oriented structure. It's very flexible to deploy, and allows for high access rates, which will be required while caching streamed twitter results.

The database will be capped to a given size. Before data insertion a quick check will be made, and if it is over the specified size, then the oldest record(s) will be popped from the database to make room for the new data.

3.5.17 High Level Data Flow

The whole application uses the same backend; the only difference is that each route (page) has different adapter code to convert data returned from the backend into the format that the frontend requires for rendering the data visualisation.

The data flow diagram on the next page is a simplified illustration, of at a very high level how data will flow through the program. In short, a streaming connection will be established with Twitter, and there will be a constant high-volume flow of tweets coming in 24/7, in real-time.

Each of these tweets will then go through the following processes:

1. Data is chunked, formatted and made safe
2. Sentiment and keywords are calculated and attached to the tweet
3. The data visualisations that display 'all-tweets' (real-time, NOT geo) will be updated
4. All non-geo-tagged tweets will be filtered out and not go any further
5. All geo-based data visualisations will be updated with the live tweet
6. Accuracy assessment will be carried out on the sentiment value. All tweets with low accuracy will be filtered out and not go any further
7. All high-accuracy geo-tagged tweets are then formatted further and inserted into the data cache

When the user makes a request for a certain topic, the process is very slightly different, as data is also fetched from the Twitter REST API, and merged with the streamed data for better results.

Some data visualisations will also require data, or processing from an additional source. This will be included in the adapter code, and hence not in the high level plan.

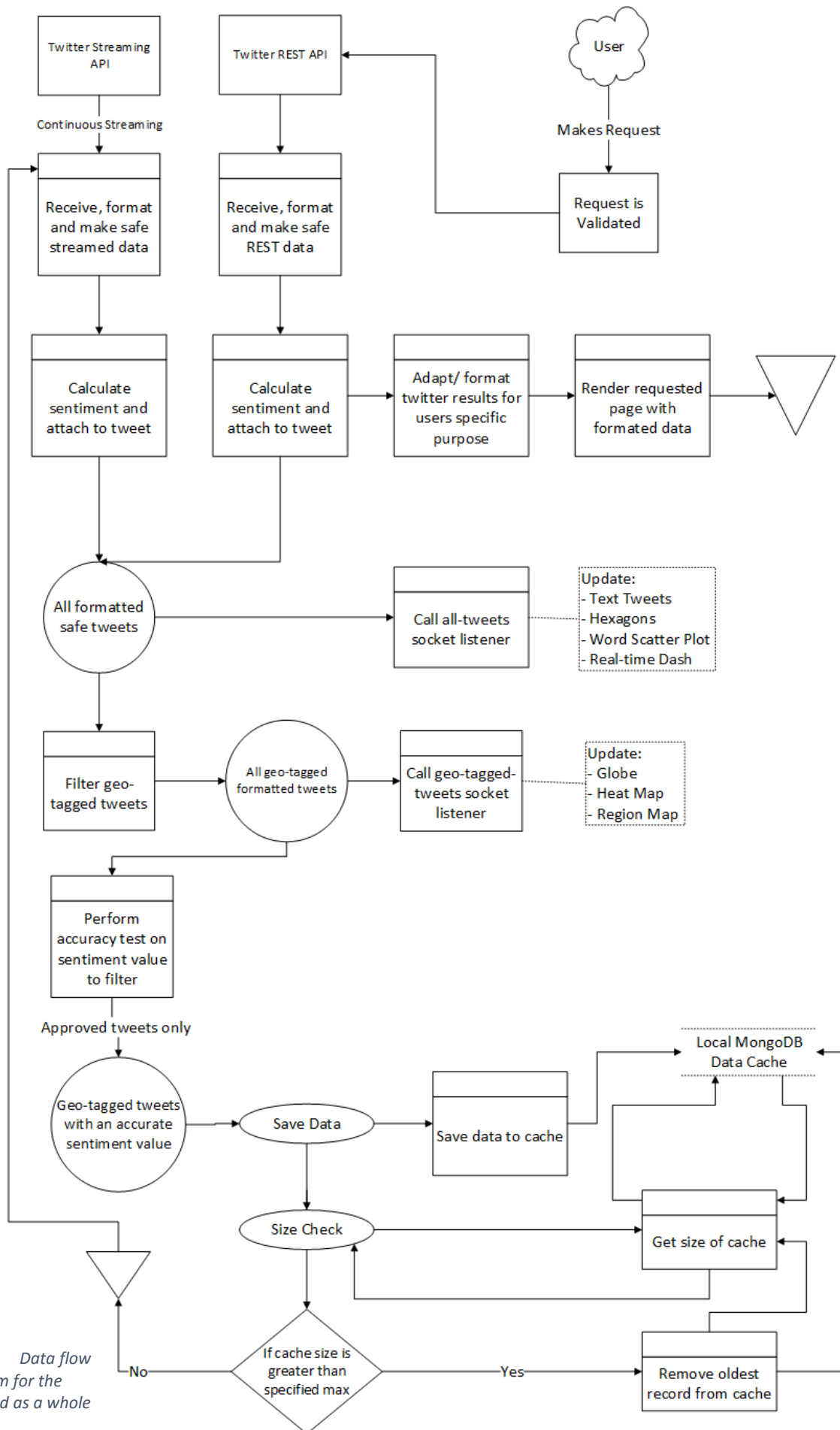


Fig 17 Data flow diagram for the backend as a whole

3.5.18 Node.js

The backend system was primarily written in Node.js. A lot of research went into choosing this, and the following section briefly summarises the findings.

Introduction to Node.js

“Node.js is a JavaScript runtime built on Chrome’s V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient” (Node.js, 2016). *Non-blocking* means that while one line of code is executing, the rest is not blocked waiting for it to finish. In traditional blocking web languages (such as Python, PHP, Ruby and Java), multiple threads of execution are used to overcome concurrency. However, in JavaScript handles it using a non-blocking event loop in a single thread. Adrian Mejia illustrates this concept clearly with a diagram similar to the one below.

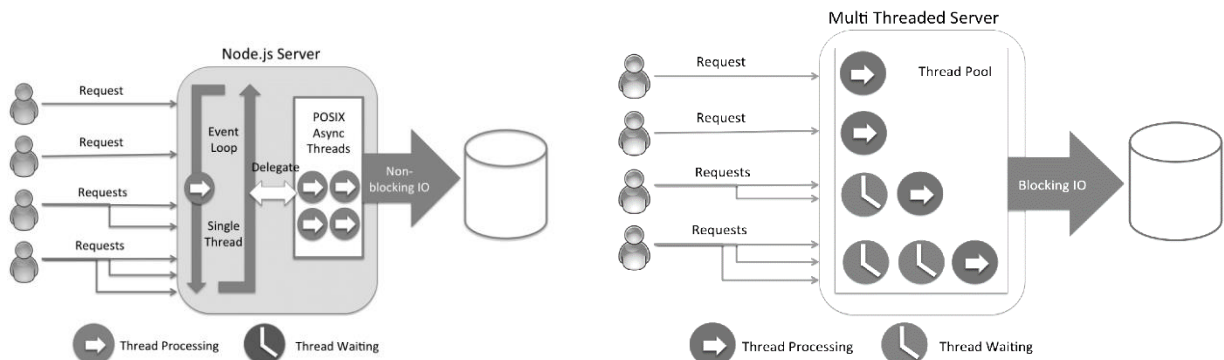


Fig 19 Node server (compared to traditional approach) Fig 18 Traditional server (compared to Node approach)

Comparison of Node.js with other popular web languages

In the journal article “Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js” (Kai Lei 2014), Node.js is compared with PHP and Python which are two of the largest players in the web industry. The experimental results in the article yield some valuable performance data, showing that PHP and Python-Web handle much less requests than that of Node.js in a certain time. Since this project shall be a very I/O intensive application, those are the results that are most relevant, and they clearly show that Node.js is up to ten times faster than traditional technologies at I/O intensive activities, and can execute many more requests per second.

Node.js for high performance web applications

Node.js is great for *developing high-performance, concurrent programs that don't rely on the mainstream multithreading approach but use asynchronous I/O with an event-driven programming model* (Stefan Tilkov 2010). Tilkov gives a details comparison between multithreading and events, the conclusion for which was that multithreading network applications can get unnecessarily complex and deadlocks can occur, as can memory leaks. However, event-driven alternatives (like Node.js) take a different approach, using callbacks to notify the system when one task has finished. Due to the large number of network, I/O and processing requests that were required for this project, Node.js was an ideal choice in terms of efficiency, speed and scalability.

Real-time data streaming in Node.js

For this project, a key requirement was that the backend could stream data live from Twitter. This is a very high volume data stream, that would need to run continuously while also carrying out other I/O intensive tasks, so the choice of technology is crucial.

The event loop used in Node.js (as opposed to threading) makes it an ideal choice for high-performance real-time server applications.

Node.js runs on V8, for more information on just how much of a significant performance difference this platform has over others see the Google I/O 2012 video: “Breaking the JavaScript Speed Limit with V8”. Further to this, another reason why Node.js can cope with high volume heavy data streams is because of its event loop model (outlined above). Xiao-Feng 2014 explains this very clearly with the aid of the diagram to the right.

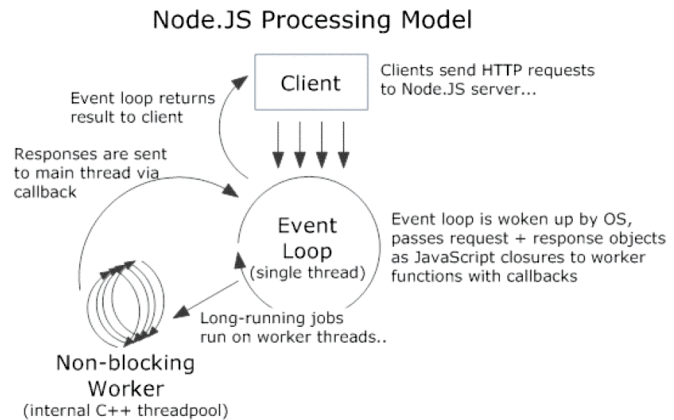


Fig 20 The Node.js processing model

Furthermore Node.js can cope very well with both basic byte chunk streams and object streams. This is outlined further in Xiao-Feng 2014’s paper on “A real-time stream system based on node.js”.

The security of Node.js

Although Node.js is by far the most scalable and efficient solution for I/O intensive applications, there are several minor security pitfalls which the developer needs to be made aware of. Firstly, the single threaded event loop in Node based architectures is a key strength in achieving scalability, although as Ojamaa 2014 pointed out, an unexpected exception will break the loop and terminate the whole program. In the same way the processing of an unexpectedly large data set can block all other connections to the application until processing is complete. Secondly, JavaScript traditionally was a basic client-side scripting language with limited functionality, for this reason many of the resources available for JavaScript are more focused on the client-side, which takes on a completely different style for the server-side. The poor quality code produced as a result of following client-side paradigms on the server-side can pose as a security risk. Furthermore Node.js is still developing very fast, and if not regularly updated then the application will be exposed to security vulnerabilities.

Overcoming the potential security pitfalls of Node.js

Several measures were taken during development to ensure this application wouldn’t be caught out by any of the vulnerabilities outlined above.

- The application will be thoroughly tested, to minimise the risk of unexpected exceptions
- A package called forever was utilised, if the application crashed for any reason (which it shouldn’t), it will be restarted automatically, and the error details written to a log
- Extensive research and upskilling will be undertaken to ensure quality server-side code is developed
- The application will be tested across many versions of the Node.js environment, and the production app will always run on the latest version of Node

4 IMPLEMENTATION

4.1 SPRINT 0 – PROJECT SETUP

4.1.1 Aims of Sprint 0

Sprint 0 covered the following user stories: TSV-T075, TSV-T076, TSV-T077, TSV-T078, TSV-T079, TSV-T080, TSV-T081, TSV-T082, TSV-T083, TSV-T084, TSV-U085, TSV-U086. (See appendix 1 for full user story's, acceptance criteria and complexity scoring).

The primary aim of sprint 0 was to set up the development environment for the project. This included the build configuration, test environment, git repository, IDE settings and creating a base directory structure – as outlined in the methodology.

4.1.2 Application meta and project configuration

NPM and app meta config

The `./package.json` file contains all npm meta data for the application, including all backend dependencies, dev dependencies, initial entry point, links to test scripts, author info etc...

The following code snippet is an extract from the file:

```
{
  "name": "twitter-sentiment-visualisation",
  "description": "A series of data visualisations showing overall sentiment from
  Tweets by location and/or topic",
  "author": "Alicia Sykes <alicia@aliciaspykes.com>",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "start": "node ./bin/www",
    "test": "gulp test",
    "build": "gulp build",
    "config": "gulp generate-config && start notepad 'config\\src\\keys.coffee'",
    "cover": "istanbul cover node_modules/mocha/bin/_mocha --dir
  ./reports/coverage-reports"
  },
  "main": "app.js",
  "repository": {
    "type": "git",
    "url": "https://github.com/Lissy93/twitter-sentiment-visualisation"
  },
  "bugs": {
    "url": "https://github.com/Lissy93/twitter-sentiment-visualisation/issues"
  },
  "dependencies": {
    "body-parser": "^1.14.0",
    "coffee-script": "^1.9.3",
    "cookie-parser": "^1.4.0",
    .....
  }
}
```

.....

[see appendix for full file]

Bower configuration

In a similar way, a meta file was configured to keep track of all frontend dependencies. The `./bower.json` stores the identifier, URL and version number of each library, package or external resource required for the frontend, this allows for easy and reliable deployments, as outlined in the methodology.

Git configuration

The application used all global default settings for Git, so the only notable file to include is the `./gitignore` which contains the globs of directories or files that should not be pushed to git. This includes backend and frontend dependencies, private keys/ passwords, debug logs, test reports and all built/compiled files. The following is the contents of the `.gitignore` file.

```
# Packages and libraries
node_modules
bower_components

# IDE files and logs
.idea
*log
npm-debug.log.*

# Test reports and results
reports
coverage

# API keys
*/**/keys.*

# Built files
models/*.js
utils/*.js
config/*.js
routes/*.js
public/javascripts
public/stylesheets
```

Test Options

The majority of the test configuration is done either externally where a third party is utilised, or within the gulp build setup (covered later). The continuous integration test options are covered in the following Yamel file (`./travis.yml`). This basically just outlines which versions of Node.js and which platforms the application will be automatically tested on, as well as any pre-scripts.

```
language: node_js
node_js:
  - "0.12"
  - "0.10"

before_script:
  - "npm i -g mocha"

notifications:
  slack:
    on_success: change
```

Mocha also required a file that specifies any configurations that weren't the default. This is included in the `./test/mocha.opts` file:

```
--compilers coffee:coffee-script/register
--reporter mochawesome
--reporter-options reportDir=./reports/awesome-reports
--recursive
```

4.1.3 Completed Gulp Build Setup

The methodology outlined how the automated build setup would be developed and configured, as well as the technologies utilised.

Initial entry point

Each task was put in a separate file within the [./tasks](#) directory. The initial file that calls the include method, was the [./gulpfile.js](#), the following is its contents (not including the extensive in-code documentation):

```
// Include gulp and require directory module
var gulp = require('gulp');
var reqdir = require('require-dir');

reqdir('./tasks'); // Include the folders containing ALL gulp tasks

gulp.task('default', ['start']); // Default task
```

Task list

The following files each contain a single Gulp task. (underlined paths are hyperlinks)

<u>./tasks/browser-sync.js</u>	Keeps multiple browsers/devices in sync during dev
<u>./tasks/browserify.js</u>	Includes all frontend dependencies in single bundle
<u>./tasks/build.js</u>	Initiate full project build
<u>./tasks/clean.js</u>	Cleans workspace/ deletes obsolete files etc..
<u>./tasks/config.js</u>	Contains JSON of various file paths
<u>./tasks/generate-config.js</u>	Generates a blank config file, for first time use
<u>./tasks/images.js</u>	Translates images into their necessary format
<u>./tasks/nodemon.js</u>	Starts a Nodemon (auto-restart) server for development
<u>./tasks/quick-coffeescript.js</u>	Very quickly compiles all CoffeeScript to JavaScript
<u>./tasks/scripts.js</u>	Compiles, lints, minifies, pipes etc... all scripts
<u>./tasks/styles.js</u>	Compiles, lint, minifies all SASS, Less, CSS to CSS
<u>./tasks/test.js</u>	Runs all unit coverage, browser etc... tests
<u>./tasks/watch.js</u>	Watches all files for changes and runs appropriate tasks

Example task 1 – Build

This task initiates all other tasks required to build the project

```
var gulp = require('gulp');
var runSequence = require('run-sequence');

/* Clean the work space */
gulp.task('build', function (cb) {
  runSequence('clean',
    ['scripts', 'browserify', 'styles', 'images'],
    cb);
});
```

Config file

All variables, such as file source and destination paths, were put into a single config file. This made the code easier to maintain, modify and reuse. The following file contains the file paths necessary to map the source project to the destination project, and all other build variables.

```
/* Define constants */
var footerText = "\n/* (C) Alicia Sykes <alicia@aliciasykes.com> 2015      "
+   "\n/* MIT License. Read full license at: https://goo.gl/IL41QJ */\n"

module.exports = {
  SOURCE_ROOT      : "client-side-source", // Dir name for all js & css src
  DEST_ROOT        : "public",           // Folder name for the results root
  CSS_DEST_DIR_NAME : "stylesheets",     // Name of CSS directory
  CSS_SRC_DIR_NAME  : "styles",         // Name of CSS directory
  JS_FILE_NAME      : "all.min.js",     // Name of output JavaScript file
  CSS_FILE_NAME     : "all.min.css",    // Name of output CSS file
  FOOTER_TEXT       : footerText,       // Optional footer text for output files
  SCRIPT_PATHS      : [ // Paths for JavaScript files
    { src: 'client-side-source/scripts/**/*.{js,coffee}',
      dest: 'public/javascripts' },
    { src: 'client-side-source/scripts/visualisations/*.coffee',
      dest: 'public/javascripts/charts' },
    { src: 'server-side-source/models/*.coffee',
      dest: 'models' },
    { src: 'server-side-source/utils/*.coffee',
      dest: 'utils' },
    { src: 'server-side-source/config/*.coffee',
      dest: 'config' },
    { src: 'server-side-source/routes/*.coffee',
      dest: 'routes' },
    { src: 'server-side-source/api-routes/*.coffee',
      dest: 'routes' }
  ],
  SHOW_OUTPUT: true
};
```

Example task 2 – Watch

This task watches files (in an array of globs from config file above) for changes, and calls the necessary task (in separate file) when there is a change to the source.

```
var gulp = require('gulp');
var CONFIG = require('../tasks/config');

/* Configure files to watch for changes - NOT USED BY DEFAULT TASK */
gulp.task('watch', function() {

  /* For each CoffeeScript directory, watch, compile and reload */
  CONFIG.SCRIPT_PATHS.forEach(function(path) {
    gulp.watch(path.src, ['scripts']);
  });

  /* Watch build and re-bundle browserify files */
  gulp.watch(CONFIG.SOURCE_ROOT+'/**/*.{main,module}.{js,coffee}',
    ['browserify']);

  /* Watch compile and reload LESS, SASS and CSS */
  gulp.watch(CONFIG.SOURCE_ROOT+'/**/*.{css,less}', ['styles']);

});
```


Example task 3 – styles

This task handles all frontend styles

```
/* Include the necessary modules */
var gulp    = require('gulp');
var gutil   = require('gulp-util');
var gsize   = require('gulp-filesize');
var concat  = require('gulp-concat');
var less    = require('gulp-less');
var minCss  = require('gulp-minify-css');
//var cssLint = require('gulp-csslint'); // Uncomment to lint CSS
var changed = require('gulp-changed');
var footer  = require('gulp-footer');
var es      = require('event-stream');
var gulpIf  = require('gulp-if');
var argv    = require('yargs').argv;

var CONFIG  = require('../tasks/config');

var devMode = argv.dev ? true : CONFIG.SHOW_OUTPUT;

/* CSS and Less Tasks */
gulp.task('styles', function(){

    // Paths
    var cssSrcPath = CONFIG.SOURCE_ROOT + '/' + CONFIG.CSS_SRC_DIR_NAME;
    var cssResPath = CONFIG.DEST_ROOT + '/' + CONFIG.CSS_DEST_DIR_NAME;

    // Streams
    var cssFromLess = gulp.src([cssSrcPath+'/*.less']).pipe(less());
    var cssFromVanilla = gulp.src([cssSrcPath+'/*.css']);
    var excludedCss = gulp.src(cssSrcPath+'/**/*.css');
    var excludedLess = gulp.src(cssSrcPath+'/**/*.less').pipe(less());

    var concatenatedCss = es.merge(cssFromLess, cssFromVanilla)
        .pipe(concat(CONFIG.CSS_FILE_NAME));

    return es.merge(concatinatedCss, excludedCss, excludedLess)
        .pipe(changed(cssResPath))
        // .pipe(cssLint()) // Uncomment to lint CSS
        .pipe(minCss({compatibility: 'ie8'}))
        .pipe(gulpIf(devMode, gsize()))
        .pipe(footer(CONFIG.FOOTER_TEXT))
        .pipe(gulp.dest(cssResPath))
        .on('error', gutil.log);
});
```

Example Task 5 – Browserify

This script pulls all the client-side dependencies in a script together and compiles everything into a single JavaScript file, ready to be run on a browser.

```
var gulp    = require('gulp');
var browserify = require('browserify');
var source   = require('vinyl-source-stream');
var coffeeify = require('coffeeify');
var reactify  = require('reactify');
var buffer    = require('vinyl-buffer');
//var sourcemaps = require('gulp-sourcemaps'); // Uncomment to use sourcemaps
var glob     = require('glob');
var es       = require('event-stream');
var rename   = require('gulp-rename');
var footer   = require('gulp-footer');
var gutil    = require('gulp-util');
var gsize    = require('gulp-filesize');
var debowerify = require('debowerify');
```

```

var CONFIG = require('../tasks/config');

gulp.task('browserify', function () {
  glob('./'+CONFIG.SOURCE_ROOT+'/scripts/**/*-main.{js,coffee,jsx}',
function(err, files) {
  var tasks = files.map(function(entry) {
    return browserify({ entries: [entry], debug: true })
      .transform(coffeeify)
      .transform(reactify)
      .transform(debowerify)
      .bundle()
      .pipe(source(entry))
      .pipe(rename(function(filepath) {
        filepath.basename = filepath.basename.replace("-main", "");
        filepath.dirname = "";
        filepath.extname = ".bundle.js";

      })))
      .pipe(buffer())
      /* Uncomment the next two lines for source maps for debugging */
      // .pipe(sourcemaps.init({loadMaps: true}))
      // .pipe(sourcemaps.write('.'))
      .pipe(footer(CONFIG.FOOTER_TEXT))
      .pipe(gsize())
      .pipe(gulp.dest('./public/javascripts'))
      .on('error', gutil.log);
  });
  es.merge(tasks);
});

return gulp.src('*');
});

```

Example Task 6 – Quick CoffeeScript

This script very quickly, simply just compiles all CoffeeScript files in the source directories into JavaScript files, and pipes them to the result directory, bypassing any formal checking

```

var gulp = require('gulp');
var coffee = require('gulp-coffee');
var CONFIG = require('../tasks/config');

gulp.task('quick-coffeescript-watch', function(){
  gulp.watch('./**/*.coffee', ['quick-coffeescript']);
});

gulp.task('quick-coffeescript', function(){

  /* A list of file sources and destinations */
  var paths = CONFIG.SCRIPT_PATHS;

  /* For each file path, run all script tasks */
  var tasks = [];
  paths.forEach(function(path) { tasks.push(coffeeify(path.src, path.dest))});
  return (tasks); // Return all results as array of multiple streams

  /* Function applies all the tasks on the script files and returns a pipe dest
  */
  function coffeeify(srcPath, resPath){
    return gulp.src(srcPath).pipe(coffee()).pipe(gulp.dest(resPath));
  }
});

```

4.1.4 The Test Setup

The test setup matches that of the project methodology. A full example of a set of tests can be found in Appendix 4.

In short, the project has unit tests following the Mocha framework, and using the Chai assertion library, Travis CI was used to implement the automated continuous integration testing (see above, configuration section 4.1.1). The tests were primarily run using Gulp, and a summary of results were printed to the console, as well as a detailed visual report generated. Coverage results were calculated too, which showed what percentage of the code was covered by detailed unit tests, using Istanbul. Automated code reviews (using Codeacy and CodeClimate) and dependency checking (using David DM) was configured and implemented too.

Structure of the unit tests

- Every class has its own **test script**, made up of a series of describe blocks
- Every function/method from the class, has its own **describe block** within the test file, which very briefly describes what that function should do. A good quality function should only do one simple action, so describing what it does is easy.
- Every describe block is made up of a series of **it blocks**, each of which specifies a condition that must always be true for the function to be valid.
- Every it block, is made up of a series of **assertion statements** which actually test the condition specified in the it block, with various different valid, borderline and invalid values.

Every unit test file starts by: importing the desired assertion library, importing the module to test (and its private methods) and setting the current environment to test mode (if isn't already). For example:

```
expect = require('chai').expect
process.env.NODE_ENV = 'test'
removeWords = require('../index')._private
```

Example describe block:

Note: this file will be made up of many describe blocks (one for each function), and each 'it' block will actually have many assertion statements, but this is a summarised example.

See Appendix 4 for full example

```
describe 'Check that arrayifySentence() correctly transforms a string into a valid array', ()->
  it 'Should correctly turn a sentence into an array', ()->
    expect(removeWords.arrayifySentence('hello world')).eql(['hello', 'world'])
    expect(removeWords.arrayifySentence('this is a longer sentence'))
      .eql(['this', 'is', 'a', 'longer', 'sentence'])
  it 'Should normalise case', ()->
    expect(removeWords.arrayifySentence('HeLlO wOrLd')).eql(['hello', 'world'])
    expect(removeWords.arrayifySentence('JAVASCRIPT')).eql(['javascript'])
  it 'Should remove duplicates', ()->
    expect(removeWords.arrayifySentence('foo foo bar foo'))
      .eql(['foo', 'foo', 'bar', 'foo'])
    expect(removeWords.arrayifySentence('foo foo BAR Foo bAr foO bar foo'))
      .eql(['foo', 'foo', 'bar', 'foo', 'bar', 'foo', 'bar', 'foo'])
  it 'Should remove blanks', ()->
    expect(removeWords.arrayifySentence('space      blank      '))
      .eql(['space', 'blank'])
  it 'Should remove special characters', ()->
    expect(removeWords.arrayifySentence('foo ! ^&*^&^%&^^&%&^bar$$%^'))
      .eql(['foo', 'bar'])
```

4.1.5 Sprint 0 Summary

Sprint 0 Progress

The development environment is now configured as per the requirements. The following user stories were completed: TSV-T075, TSV-T076, TSV-T077, TSV-T078, TSV-T079, TSV-T080, TSV-T081, TSV-T082, TSV-T083, TSV-T084, TSV-U085, TSV-U086. The next stage from here will be to develop the base application.

Sprint 0 Review

This sprint was considerably more work than estimated in the sprint planning. Configuring Gulp to do such a wide variety of tasks required researching the most efficient way to manage data streams, as well as a lot of upskilling in various technologies that are needed to bring everything together. Setting up the basic unit test environment meant that some dummy tests had to be written, again this was not factored into the time plan. Despite the large workload, all requirements were completed on time, and the development environment features a very high quality build setup.

Sprint 0 Retrospective

Start Doing

- More realistic time estimates

Stop Doing

- Spending too much time on stories which have been underestimated

Continue Doing

- In-code documentation
- Well-structured files
- Following user stories and acceptance criteria closely
- Unit testing
- Completing all tasks on schedule

4.2 SPRINT 1 – CREATE BASE APPLICATION AND DATABASE SCHEMA

The bulk of the work for sprint 1, was setting up the basic frontend layout, navigation structure, configuration of routing engine, and integrating the build setup in sprint 0 with the new application.

As well as this the source and final directory structure was determined, and is as follows:

4.2.1 Directory Structure

Source Application:

```
client-side-sauce
  /graphics
  /scripts
  /styles
server-side-sauce
  /api-routes
  /config
  /models
  /routes
  /utils
views
  /components
  /pages
  /sections
dev-tasks
```

Production Application:

```
config
  /app-config.js
  /keys.js
models
  /Tweet.js
public
  /data
  /images
  /javascripts
  /stylesheets
  /favicon.ico
routes
utils
views
```

4.2.2 Database Schema

The following code shows that the Tweet object, contains the actual Tweet (*body*), timestamp (*dateTime*), sentiment and location object

```
schema = new mongoose.Schema
  body      : String
  dateTime  : String
  sentiment : Number
  location  : type : Object , "default" : {},
  capped: { size: 491520, max: 1500, autoIndexId: true
```

4.2.3 Sprint 1 Summary

Sprint 1 Progress

The base application is now created, this included the file structure, database schema and implementing the build environment from sprint 0. The following user stories were completed: TSV-A001, TSV-A002, TSV-A003, TSV-A004, TSV-U088, TSV-U089, TSV-U090, TSV-U091. However, TSV-U087 which involved setting up data stubs was not completed. This was descoped since it is no longer necessary to be included in the main application, as all the modules should have their own data stub tests.

Sprint 1 Review

Sprint 1 had much more realistic time estimates than sprint 0. However, there may need to be a lot of changes as the application develops. One story was descoped, however everything else was completed.

Sprint 1 Retrospective

Start Doing

- Thinking ahead, to reduce the amount of changes required later
- Following an ordered structure

Stop Doing

n/a

Continue Doing

- Documentation

4.3 SPRINT 2 – ALL TWEET HANDLING BACKEND MODULES

Following the methodology, two tweet handling node modules were developed, tested, documented and published. One streams tweets in real-time, the other fetches a set of tweets based on given parameters. Links to these modules can be found below:

4.3.1 Twitter Handling Modules Developed and Published

Name of Module	Fetch-tweets
Short Description	A simple to use, feature-rich, tested node module that fetches tweets from Twitter based on topic, location, timeframe or combination
Aims	<ul style="list-style-type: none">- To pull Tweets from Twitter containing a given word or hash tag- To have the option to fetch Tweets also by location or time frame or a combination- To return a clean well formatted result
Download Page	https://www.npmjs.com/package/fetch-tweets
Source Code	https://github.com/Lissy93/fetch-tweets
CI Tests	https://travis-ci.org/Lissy93/fetch-tweets
Status	Complete and published with all tests passing

Table 7 Summary of fetch-tweets module

Name of Module	Stream-tweets
Short Description	A Node module that uses the Twitter API to stream live Tweets from a specific topic or location in real-time
Aims	<ul style="list-style-type: none">- Pull real-time Tweets from Twitter- Return only fully formed Tweet objects, not chunks- Format results nicely- Provide optional parameters to stream by keyword, location and more
Download Page	https://www.npmjs.com/package/stream-tweets
Source Code	https://github.com/Lissy93/stream-tweets
CI Tests	https://travis-ci.org/Lissy93/stream-tweets
Status	Complete and published with all tests passing

Table 8 Summary of stream-tweets module

Once the base modules were complete, a few further scripts were written using these modules, within the application (as opposed to separate published modules) to do more specific actions regarding the fetching of tweets.

Below are two scripts as an example; the first, fetches tweets and assigns a sentiment value to each tweet, before returning the results. The second script extends this functionality, and also asynchronously fetches a series of tweets based on many search terms, and then initiates a call back once all results are returned. Both use the fetch-tweets module (see above).

4.3.2 In-project further twitter actions

server-side-scripts/utis/fetch-sentiment-tweets.coffee

```
# Include relevant node modules
FetchTweets = require 'fetch-tweets'
sentiment    = require 'sentiment-analysis'
removeWords  = require 'remove-words'
twitterKey   = require('../config/keys').twitter
Tweet        = require '../models/Tweet'

module.exports = (searchTerm, callback) ->

  if searchTerm == '' then Tweet.getAllTweets (results) ->
    format results, callback

  else (new FetchTweets twitterKey).byTopic searchTerm, (results) ->
    format results, callback

format = (results, callback) ->

  # Assign Sentiments
  for tweet in results then tweet.sentiment = sentiment tweet.body

  # Assign keywords
  for tweet in results then tweet.keywords = removeWords tweet.body

  # Find average sentiment
  total = 0
  for tweet in results then total += tweet.sentiment
  averageSentiment = total / results.length
  averageSentiment = Math.round(averageSentiment*100)/100

  # Done, call callback with results and sentiment average
  callback results, averageSentiment
```

server-side-scripts/utis/async-tweets.coffee

```
fetchSentimentTweets = require '../utis/fetch-sentiment-tweets'
q = require 'q'

makeTweetPromiseArr = (searchTerm) ->
  deferredTweetResults = q.defer()
  fetchSentimentTweets searchTerm, (results) -> # Fetch all data for one Tweet
    deferredTweetResults.resolve(results)
  deferredTweetResults.promise

makeRequests = (searchTerms, completeAction) ->
  results = []
  promises = []
  for term in searchTerms then promises.push makeTweetPromiseArr term
  q.all(promises).spread -> # When all the twitter promises have returned
    for argument, index in arguments
      results.push searchTerm: searchTerms[index], tweets: argument
    completeAction results # Done!

module.exports = makeRequests
```


4.3.3 Sprint 2 Summary

Sprint 2 Progress

The aim of sprint 2, was to complete all Twitter handling operations. The following user stories were completed during this sprint: TSV-E021, TSV-E022, TSV-E023, TSV-E024, TSV-E025, TSV-V093, TSV-V094. This included developing two open-source modules for fetching and streaming tweets, and also several scripts to do more advanced or specific operations (like multiple async requests) with tweets.

Sprint 2 Review

Everything was completed within time, and the estimates proved very accurate. No additional unplanned functionality was added. A lot of documentation was written and published.

Sprint 2 Retrospective

Start Doing

- Better unit tests
- More planning ahead
- Better CoffeeScript style

Stop Doing

n/a

Continue Doing

- Modularising code
- Realistic time and complexity estimates

4.4 SPRINT 3 – ALL LOCATION AND UTILITY BACKEND MODULES

As outlined in the methodology, each utility script would be developed and published as a node module. Below are the three location-based modules that were coded and then implemented into the application.

Name of Module	Find-region-from-location
Short Description	A lightweight node module that, given a latitude and longitude calculates which region that point belongs in, without any external requests
Aims	<ul style="list-style-type: none"> - To return valid place name, region code and ISO format for any given valid latitude and longitude - To return the closest region if latitude and longitude falls outside any region (e.g. on coast of Scotland) - To be fast, light-weight and efficient
Source Code	https://github.com/Lissy93/find-region-from-location
Status	Complete and published

Table 9 Summary of find-region-from-location module

Name of Module	Place-lookup
Short Description	A lightweight Node.js module to get the latitude and longitude for any fuzzy place name using the Google Places API
Aims	<ul style="list-style-type: none"> - To return valid latitude and longitude data for any place - To find closest match of a place if type of no exact matches - To find most likely place if multiple results returned - To be fast, light-weight and efficient
Download Page	http://npmjs.com/package/place-lookup
Source Code	https://github.com/Lissy93/place-lookup
Status	Complete and published

Table 10 Summary of place-lookup module

Name of Module	Tweet-location
Short Description	Calculates the location from geo-tagged Tweets using the Twitter Geo API
Aims	<ul style="list-style-type: none"> - To return a latitude and longitude object capable of being plotted on a Google Map for a given Tweet - Additionally optionally return an object of place information for other services
Download Page	https://www.npmjs.com/package/tweet-location
Source Code	https://github.com/Lissy93/tweet-location
Status	Complete and published with all tests passing

Table 11 Summary of tweet-location module

4.4.1 Sprint 3 Summary

Sprint 3 Progress

Other than the sentiment analysis module, all other standalone modules are now completely finished, tested, documented and published. They are ready to be integrated into the application in sprint 5. User stories TSV-X097, TSV-X098, TSV-X099, TSV-Z100, TSV-E025 and TSV-V094 were all completed fully.

4.5 SPRINT 4 – SENTIMENT ANALYSIS MODULE

This quite possibility, is the most important component of the project, so a lot of attention was paid to getting it as accurate, stable and as fully tested as possible.

Name of Module	Sentiment-analysis
Short Description	Uses the AFINN-111 word list to calculate overall sentiment of a sentence
Aims	<ul style="list-style-type: none">- To very quickly calculate the overall sentiment of a sentence or string of text- No additional network requests required- Efficient algorithm- Thorough testing
Download Page	https://www.npmjs.com/package/sentiment-analysis
Source Code	https://github.com/Lissy93/sentiment-analysis
CI Tests	https://travis-ci.org/Lissy93/sentiment-analysis
Status	Complete and published with all tests passing

Table 12 Summary of sentiment-analysis module

Since the sentiment-analysis module was published, 6 months ago – it has had over 1000 individual downloads, and rising. It's thorough test strategy means so far there have been zero bugs reported, and the clear documentation makes it an attractive choice for developers.

Another, smaller module was also developed and published during this sprint. The remove-words node module extracts keywords for a given string and returns them in the form of an array. It uses no external resources, and is self-learning. It will have many uses in this application, both on the backend and frontend.

Name of Module	Remove-words
Short Description	Removes all non-keywords from a string
Aims	<ul style="list-style-type: none">- To allow an array of words to be removed from a string- By default all pronouns, conjunctions,
Download Page	https://www.npmjs.com/package/remove-words
Source Code	https://github.com/Lissy93/remove-words
CI Tests	https://travis-ci.org/Lissy93/remove-words
Status	Complete and published with all tests passing

Table 13 Summary of remove-words module

4.5.1 Sprint 4 Summary

Sprint 4 Progress

The sentiment analysis component of the application obviously is very important, and now that the sprint is complete I am confident that this module is developed to a suitably high quality. User stories: TSV-D015, TSV-D016, TSV-D017, TSV-D018, TSV-D019 and TSV-D020 were all completed fully.

Sprint 4 Retrospective

Start Doing

- More content could be covered in this sprint. Story points were over-estimated.

Continue Doing

- Very thoroughly tested module, if the whole app was tested like this, it would be close to bug-free!

4.6 SPRINT 5 – INTEGRATE BACKEND MODULES INTO APPLICATION

Once all the key backend modules were developed they could be integrated into the application.

This was done, first by running each of their installation commands (e.g. `npm install sentiment-analysis --save`) which downloads the files into the `./node_modules` directory, and the `--save` flag, adds a dependency for each module into the `./package.json` file. Next the module is imported into the script(s) that need it (e.g. `var sentimentAnalysis = require('sentiment-analysis');`). Once it is imported the module can be used as indented (e.g. `console.log(sentimentAnalysis('Have a great day!'))`; will output `+0.2` as the positive sentiment value)

Several backend utility scripts were developed to do specific tasks within the application. The following class is an example of one of these utility scripts, it was reused in the backend for all geo-based data visualisations (heat map, 3d globe and regional map). It combines the Twitter API (using `fetch-tweets` module, above) with the Google Places API (using the `place-lookup` module, above) and then calculates sentiment and extracts keywords for each tweet (using the `sentiment-analysis` and `remove-words` modules, above).

```
# Include relevant node modules
FetchTweets = require 'fetch-tweets'
placeLookup = require 'place-lookup'
sentiment = require 'sentiment-analysis'
removeWords = require 'remove-words'
q = require 'q'

class GetGeoSentimentTweets

  constructor: (@twitterApiKeys, @placesApiKey) ->

  # Function fetches Tweets from Twitter API, returning a deferred promise
  fetchFromTwitter = (query, twitterApiKeys) ->
    fetchTweets = new FetchTweets(twitterApiKeys)
    deferredTweets = q.defer()
    fetchTweets.byTopic query, (results) ->
      deferredTweets.resolve(results)
    deferredTweets.promise

  # Fetches place data (lat and long) from Google places api, return promise
  findPlaceInfo = (locationObject, placesApiKey) ->
    deferredLocation = q.defer()
    if locationObject.place_name!=""
      placeLookup locationObject.place_name, placesApiKey, (placeData)->
        deferredLocation.resolve(placeData)
    else deferredLocation.resolve({error:'no place available'})
    deferredLocation.promise

  # Main
  go: (searchQuery, completeAction) ->
    placesApiKey = @placesApiKey
    fetchFromTwitter(searchQuery, @twitterApiKeys).then (twitterResults) ->
      promises = [] # array of google places promises
      for tweet in twitterResults
        promises.push findPlaceInfo tweet.location, placesApiKey
      q.all(promises).spread -> # When all the places promises have returned
        for tweet, index in twitterResults
          tweet.location = arguments[index] # Attach new location to Tweet
          tweet.sentiment = sentiment tweet.body # Attach sentiment to Tweet
          tweet.keywords = removeWords tweet.body # Attach keywords to Tweet
        completeAction twitterResults # Done!

module.exports = GetGeoSentimentTweets
```

4.7 SPRINT 6 – GEO SENTIMENT DATA VISUALISATIONS

This sprint involved developing the three location-based sentiment data visualisations; the heat map, 3d globe and regional map. The backend of all these visualisations used the script above as part of the data flow.

4.7.1 The Heat Map

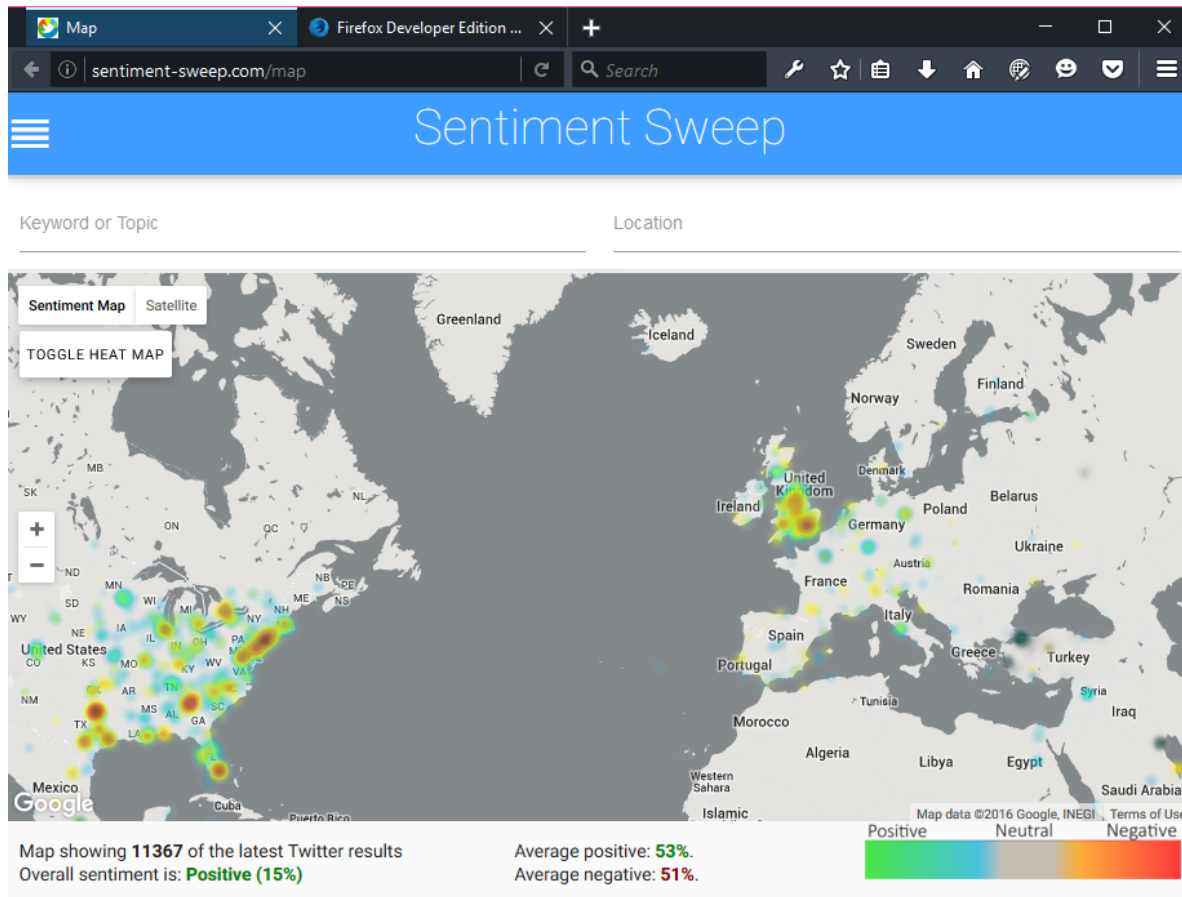


Fig 22 Screenshot of heat map

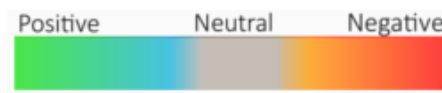
Number of tweets and average sentiment

This is displayed in the lower-right corner. It quickly shows the user how many tweets are on the map.

Map showing 28164 of the latest Twitter results
Overall sentiment is: Positive (14%)

Colour key

This indicates to the user, what the different colours represent. (green= very positive, blue = slightly positive, grey = neutral, orange = slightly negative, red = very negative).



Zooming

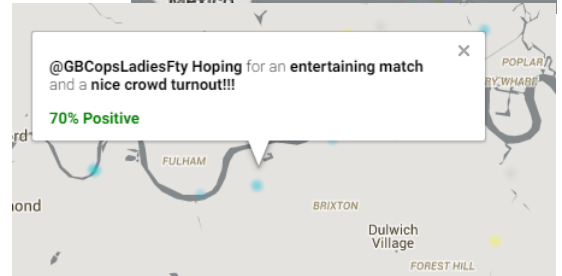
The user can zoom to an area, either by using the on-screen buttons, mouse wheel or double tapping. The amount of data displayed per area increases on zoom.

If the user clicks a dense heat area, the map will automatically zoom to that area.



Reading individual tweets

Once the user has zoomed to or clicked an area, they can click a location to read the tweet. Keywords will be highlighted in bold, and are clickable. The sentiment score (percentage and positive or negative) is displayed in the bottom left. This dialog disappears once the user clicks another part of the map.

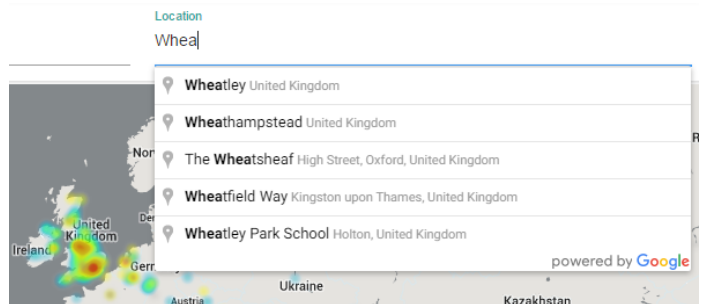


Searching by topic

The user can search by topic, using the top left input box. They press enter to submit the search term and the map will re-render.

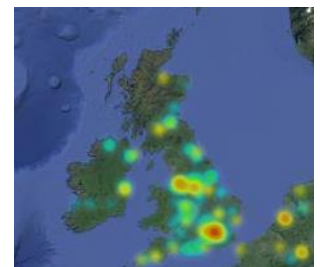
Search by location

As specified in the requirements, the user can also search the sentiment heat map by location. There is an input field, with a dropdown that uses Google's places API, very similar to that on Google maps.



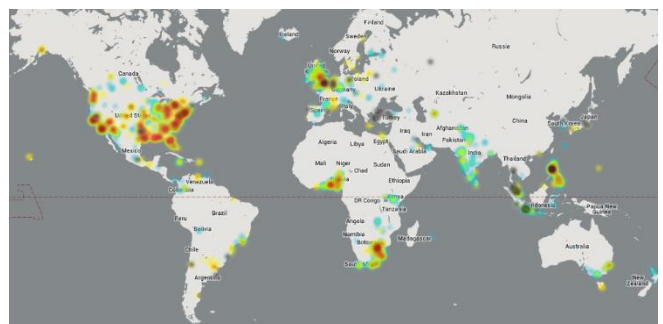
Satellite view

The map also has a satellite view option, which makes use of Google Earth's satellite tiles. The user can enable this mode with the toggle button in the top left corner. The rest of the maps functionality will continue to work as before, and all data will be preserved.



Hide old heat map data

There is also the option to toggle the heat map data, this will hide the old data, and show only real-time results as they come in. This functionality demonstrates just how many location based tweets there are every second.



4.7.2 3D Sentiment Globe

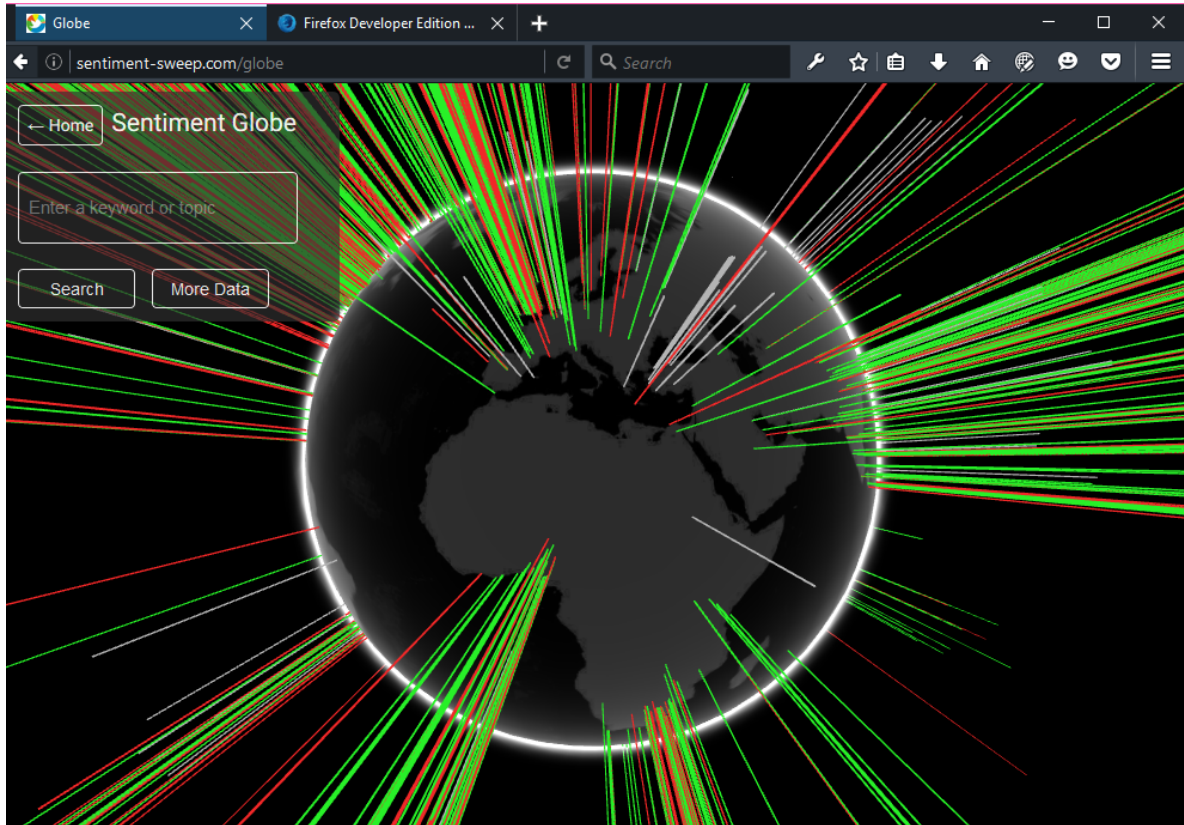


Fig 23 Screen shot of 3D globe

About the globe

Each line represents a sentiment area. Green represents positive, red negative and grey neutral. The height of the bar illustrates both volume of tweets and strength of sentiment (e.g. 2 x 80% positive tweets will have a greater height than 3 x 10% positive). Although the numbers are obviously much larger).

The user can interact with the globe, rotating it and zooming in with their pointing device (mouse) or touch screen.

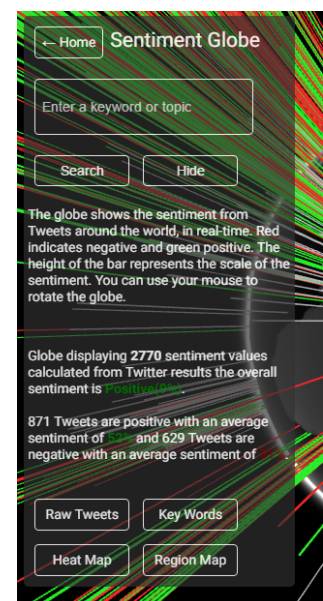
The globe is real-time, and when a new tweet comes in it is displayed immediately. This makes it very cool to watch.

Viewing more information about the data on the globe

The user can click or tap “See More” to read about what the globe is showing, as well as the latest sentiment stats and links to other geo data visualisations.

Entering a custom search term

In the same way as on the map, the user can enter a search term. This can be a topic, location, person, political party, sports team or anything. Fresh twitter data will be fetched, and the globe will be re-rendered.



4.7.3 Regional Map

Often it is useful to get a clear snapshot of which regions are positive or negative about a specific topic, and for this the heat map and globe are too detailed. The aim of the regional map was to show the top positive and negative regions (both national and international) for a specific search term (or overall).

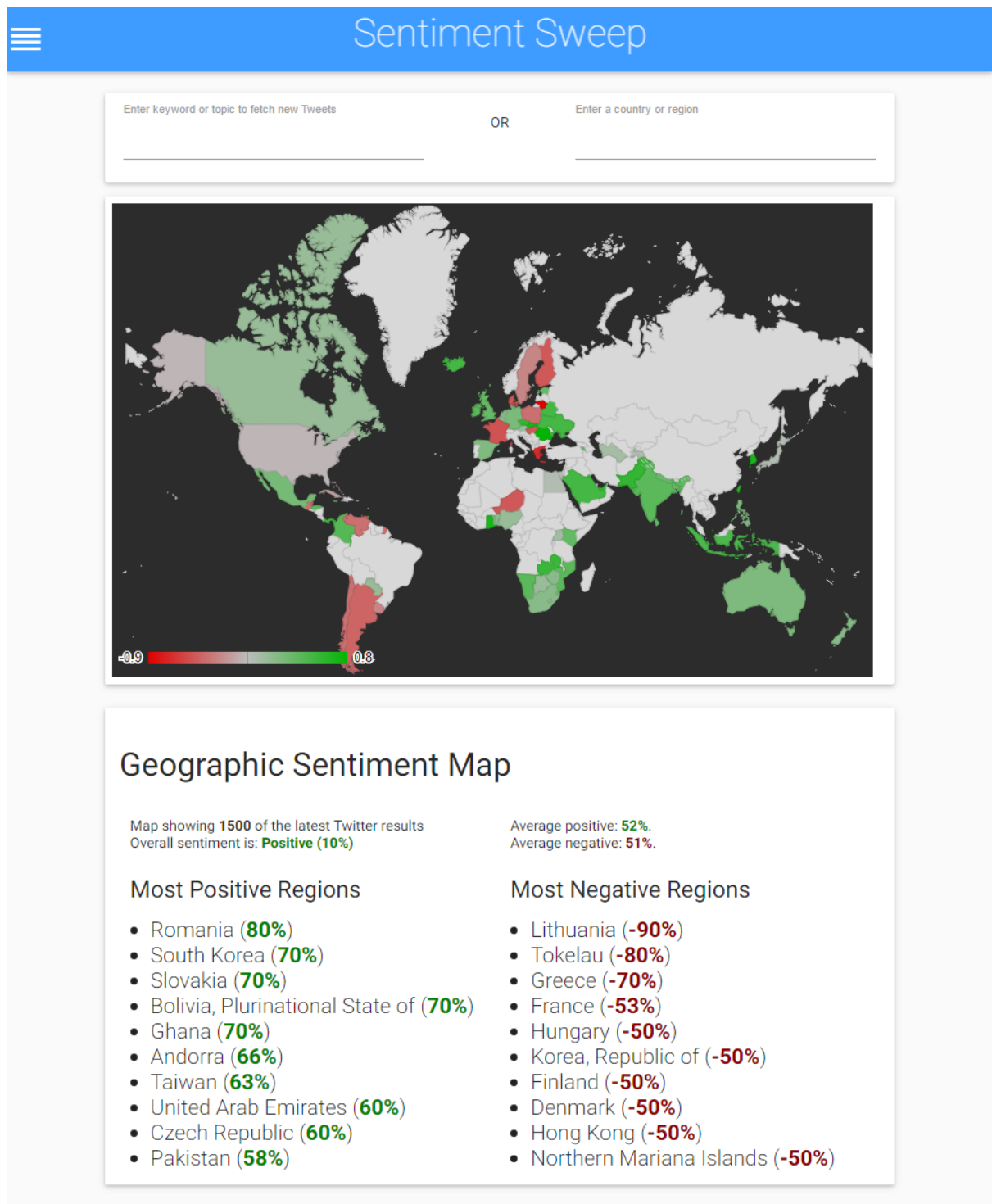


Fig 24 Screen shot of regional map

As the above print screen shows, each region has a colour based on sentiment. The user can hover over a region to view exact sentiment score, and click a region to zoom to sub-regions. The user can enter either a custom search term, or a region name. At the bottom of the page there is a list of most positive and negative regions for the users search term.

CoffeeScript D3.js Source Code for Scatter Plot

```
d3 = require 'd3/d3.min.js'
tipsy = require 'tipsy/index.js'

mainPage = 'word-scatter-plot'
pageControls = require '../page-controls-module.coffee'
pageControls.setMainPage mainPage

# set the stage
margin = {t: 30, r: 20, b: 20, l: 40}
w = 600 - (margin.l) - (margin.r)
h = 500 - (margin.t) - (margin.b)

biggestFreq = 10
for e, i in wordData
  if e.freq > biggestFreq then biggestFreq = e.freq
  ran = Math.round(Math.random()*10)/100
  wordData[i].sentiment = wordData[i].sentiment + ran

x = d3.scale.linear().range([0, w/4, w/2, (w/4)*3, w])
y = d3.scale.linear().range([h - 60, 0])

scaleColors = ["#a50026", "#d73027", "#fdae61", "#B4B4B4",
               "#a6d96a", "#1a9850", "#006837"]

color = d3.scale.linear()
  .domain([-0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6])
  .range(scaleColors)

svg = d3.select('#scatter-words')
  .append('svg')
  .attr('width', w + margin.l + margin.r)
  .attr('height', h + margin.t + margin.b)

# set axes, as well as details on their ticks
xAxis = d3.svg.axis().scale(x)
  .ticks(20).tickSubdivide(true).tickSize(6, 3, 0).orient('bottom')
yAxis = d3.svg.axis().scale(y)
  .ticks(20).tickSubdivide(true).tickSize(6, 3, 0).orient('left')

# group that will contain all of the plots
groups = svg.append('g')
  .attr('transform', 'translate(' + margin.l + ', ' + margin.t + ')')

# array of the sentiments, used for the legend
sentiments = ['Positive', 'Neutral', 'Negative']

x0 = Math.max(-d3.min(wordData, (d) -> d.freq), d3.max(wordData, (d) -> d.freq))
x.domain [0, 10, 20, 50, biggestFreq]
y.domain [-0.6, 0.6]

# style the circles, set their locations based on data
circles = groups.selectAll('circle')
  .data(wordData)
  .enter()
  .append('circle')
  .attr('class', 'circles')
  .attr(
    cx: (d) -> x + d.freq
    cy: (d) -> y + d.sentiment
    r: 8
    id: (d) -> d.text
  )
  .style('fill', (d) -> color d.sentiment)
  .style('opacity', '0.6')

# what to do when we mouse over a bubble
mouseOn = ->
  circle = d3.select(this)
  # transition to increase size/opacity of bubble
  circle.transition()
    .duration(800)
    .style('opacity', 1)
    .attr('r', 16).ease 'elastic'

# append lines that will be used to show the precise data points.
# translate their location based on margins
svg.append('g')
  .attr('class', 'guide')
  .append('line')
  .attr('x1', circle.attr('cx'))
  .attr('x2', circle.attr('cx'))
  .attr('y1', +circle.attr('cy') + 26)
  .attr('y2', h - (margin.t) - (margin.b))
  .attr('transform', 'translate(40,20)')
  .style('stroke', circle.style('fill'))
  .transition()
```

```

    .delay(200)
    .duration(400)
    .styleTween 'opacity', -> d3.interpolate 0, .5

    svg.append('g')
      .attr('class', 'guide')
      .append('line')
      .attr('x1', +circle.attr('cx') - 16)
      .attr('x2', 0)
      .attr('y1', circle.attr('cy'))
      .attr('y2', circle.attr('cy'))
      .attr('transform', 'translate(40,30)')
      .style('stroke', circle.style('fill'))
      .transition().delay(200).duration(400)
      .styleTween 'opacity', -> d3.interpolate 0, .5

    # func to move mouseover item to front of SVG stage, if another dot overlaps
    d3.selection::moveToFront = -> @each -> @parentNode.appendChild this
    circle.moveToFront()

    # what happens when we leave a bubble?
    mouseOff = -> circle = d3.select(this)

    # go back to original size and opacity
    circle.transition()
      .duration(800)
      .style('opacity', .4)
      .attr('r', 8)
      .ease 'elastic'
    # fade out guide lines, then remove them
    d3.selectAll('.guide').transition().duration(100).styleTween('opacity', ->
      d3.interpolate .5, 0
    ).remove()

    # run the mouseon/out functions
    circles.on 'mouseover', mouseOn
    circles.on 'mouseout', mouseOff

    # tooltips (using jQuery plugin tipsy)
    circles.append('title').text (d) -> d.text
    $('.circles').tipsy gravity: 's'

    circles
      .attr('class', 'tooltipped')
      .attr('data-position', 'bottom')
      .attr('data-delay', '50')
      .attr('data-tooltip', (d) -> d.text)

    # the legend color guide
    legend = svg.selectAll('rect')
      .data(sentiments)
      .enter()
      .append('rect')
      .attr(
        x: (d, i) -> 40 + i * 80
        y: h
        width: 25
        height: 12)
      .style('fill', (d) ->
        if d == 'Positive' then return color 0.5
        if d == 'Negative' then return color -0.5
        else return color 0)

    # legend labels
    svg.selectAll('text').data(sentiments).enter().append('text').attr(
      x: (d, i) -> 40 + i * 80
      y: h + 24).text (d) -> d

    # draw axes and axis labels
    svg.append('g')
      .attr('class', 'x axis')
      .attr('transform', "translate(#{margin.l}, #{(h - 60 + margin.t)})")
      .call xAxis

    svg.append('g')
      .attr('class', 'y axis')
      .attr('transform', "translate(#{margin.l}, #{margin.t})")
      .call yAxis

    svg.append('text')
      .attr('class', 'x label')
      .attr('text-anchor', 'end')
      .attr('x', w + 50)
      .attr('y', h - (margin.t) - 5)
      .text 'Frequency'

    $(document).ready -> $('.tooltipped').tooltip delay: 50

```

Server Side code for preparing the keyword data for both data visualisations

```
Tweet = require './models/Tweet' # The Tweet model
MakeSummarySentences = require './utils/make-summary-sentences'
removeWords = require 'remove-words'
sentimentAnalysis = require 'sentiment-analysis'
FetchTweets = require 'fetch-tweets'

# API keys
twitterKey = require('./config/keys').twitter

fetchTweets = new FetchTweets twitterKey

class FormatWordsForCloud

  # Converts ordinary Tweet array to suitable form for word cloud
  formatResultsForCloud= (twitterResults, allWords = false) ->
    results = []
    tweetWords = makeTweetWords twitterResults
    for word in tweetWords
      sent = sentimentAnalysis word
      if allWords or sent != 0
        f = results.filter((item) -> item.text == word)
        if f.length == 0 then results.push(text: word, sentiment: sent, freq: 1)
        else for res in results then if res.text == word then res.freq++
    results

  findTopWords= (cloudWords) ->
    posData = cloudWords.filter (cw) -> cw.sentiment > 0
    negData = cloudWords.filter (cw) -> cw.sentiment < 0
    neutData = cloudWords.filter (cw) -> cw.sentiment == 0

    posData.sort (a, b) -> parseFloat(a.freq) - parseFloat(b.freq)
    posData = posData.reverse().slice(0,5)
    negData.sort (a, b) -> parseFloat(a.freq) - parseFloat(b.freq)
    negData = negData.reverse().slice(0,5)
    neutData.sort (a, b) -> parseFloat(a.freq) - parseFloat(b.freq)
    neutData = neutData.reverse().slice(0,5)

    {topPositive: posData, topNegative: negData, topNeutral: neutData}

  findTrendingWords = (cloudWords) ->
    cloudWords.sort (a, b) -> parseFloat(a.freq) - parseFloat(b.freq)
    cloudWords = cloudWords.reverse().slice(0,10)

  # Make a paragraph of keywords
  makeTweetWords = (twitterResults) ->
    para = ''
    for tweet in twitterResults then para += tweet.body + ' '
    removeWords para, false

  # Merge two sets of results
  mergeResults = (res1, res2) -> res1.concat res2

  # Make sentence description for map
  makeSentence = (data, searchTerm) ->
    (new MakeSummarySentences data, searchTerm).makeMapSentences()

  # Calls methods to fetch and format Tweets from the database
  renderWithDatabaseResults: (cb) ->
    Tweet.getAllTweets (tweets) ->
      cloudData = formatResultsForCloud(tweets)
      sentence = makeSentence(tweets, null)
      sentence.topWords = findTopWords cloudData
      cb cloudData, sentence

  # Calls methods to fetch fresh Twitter, sentiment, and place data
  renderWithFreshData: (searchTerm, cb) ->
    fetchTweets.byTopic searchTerm, (webTweets) ->
      Tweet.searchTweets searchTerm, (dbTweets) -> # Fetch matching db results
        data = mergeResults webTweets, dbTweets
        cloudData = formatResultsForCloud(data, true)
        sentence = makeSentence(data, searchTerm)
        sentence.topWords = findTopWords cloudData
        sentence.trending = findTrendingWords cloudData
        cb cloudData, sentence

  findTrends: (tweets) ->
    findTopWords formatResultsForCloud tweets, true

fwfc = new FormatWordsForCloud()
module.exports.getFreshData = fwfc.renderWithFreshData
module.exports.getDbData = fwfc.renderWithDatabaseResults
```

4.9 SPRINT 8 – TIMELINE, TRENDING AND COMPARISON CHARTS

The aim of this sprint was to use the existing backend to create a series of data visualisations comparing sentiment against other factors (time, other topics, local trends). It covered user stories: TSV-J043, TSV-J044, TSV-J045, TSV-J046, TSV-O057, TSV-O058, TSV-P059, TSV-P060, TSV-P061, TSV-P062.

During this sprint everything went smoothly, mainly because the backend was already created and tested at this point, and the infrastructure for creating similar frontend charts was already in place after sprint 7.

4.9.1 Sentiment Timeline

The timeline shows sentiment (y-axis) over time(x-axis), either overall or for a specific search term. It aims to give users an overview of what times of day the nation is more or less positive about a given topic. The user can also hover over a specific point in time to view exact values, and more data.

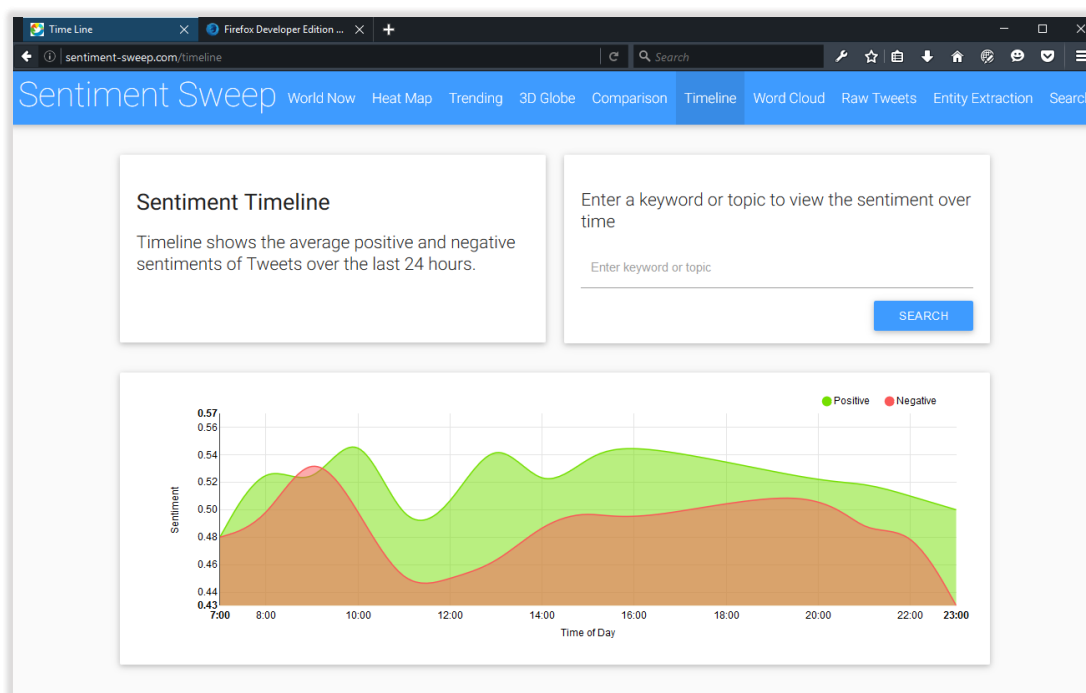


Fig 27 Screen shot of time line

A decision was made to use nvD3.js for this chart as opposed to developing it from scratch in D3. This was because it's quite a generic area line chart, and there seemed little point in rewriting something that already existed. The chart of course, still needed to be configured, but it was possible to do this in just 40 lines of CoffeeScript.

Another advantage of NVD3 is that it's fully responsive, out-of-the-box. Whereas in plain D3 it is a lot of work to implement this functionality. NVD3 can't cope with large amounts of data practically well, though this isn't an issue here, as there are a maximum of (just) 48 data points on this graph at any given time.

4.9.2 Trending

The aim of this screen was to show the sentiments of the topics currently trending on Twitter worldwide, and in the user's local area.

Screenshot of Worldwide trending screen

The trending screen is made up of three key areas. Firstly, an input field where the user can enter a location (country, region, postcode, street address – anything!).

The second part of the page shows a list of the topics that are currently trending in that area (in this screenshot, it is the whole world). Each topic is colour coded according to sentiment (green, grey or red). Topics are also clickable and will take the user to the search screen giving them more information.

Finally, there is a bubble chart. It shows each trend as a bubble, where the size indicates volume of tweets, and the colour shade indicates sentiment. Again this chart is interactive, and the user can click a bubble.

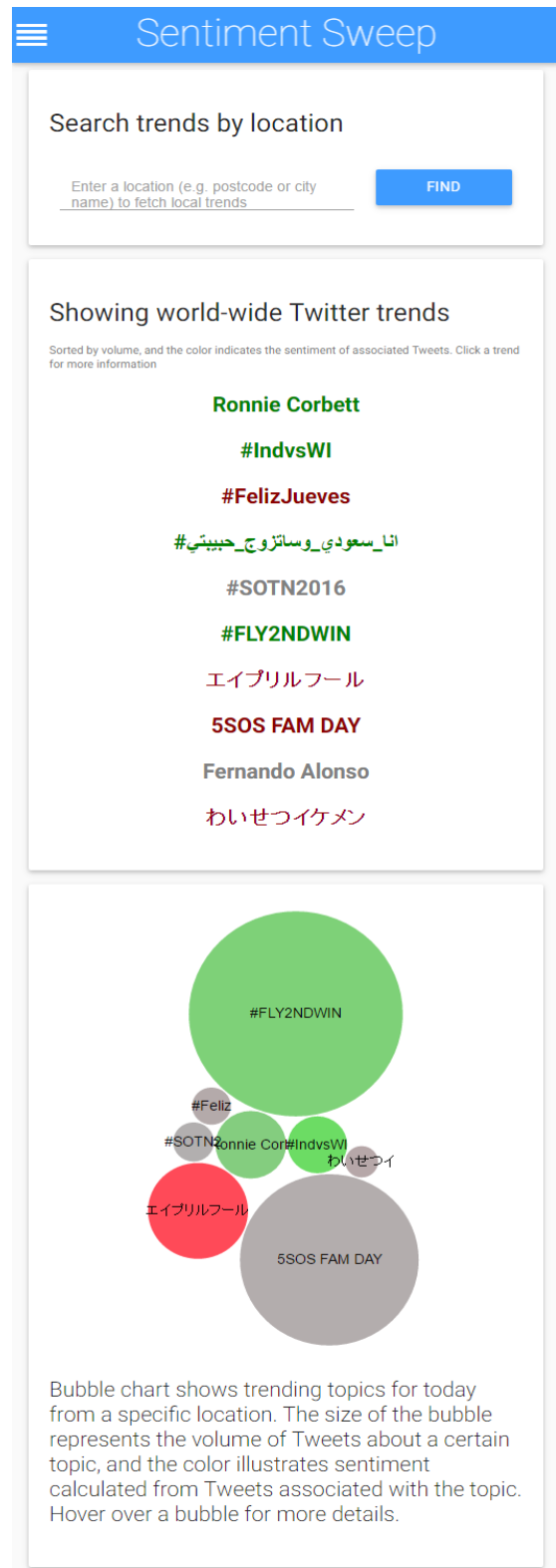


Fig 28 Screen shot of trending 1

Screenshot of the Local Trending Screen

The layout of this screen, visually is very similar. The backend however is very different (see next page).

Where the user enters their location, the place-lookup module (developed in sprint 2), finds the closest match and its latitude and longitude.

In this example, the user has entered "ox37qa" which is their home post code. The place-lookup module has updated it to "Grays Road, Oxford, UK". This reassures the user, that trends are from the correct place.

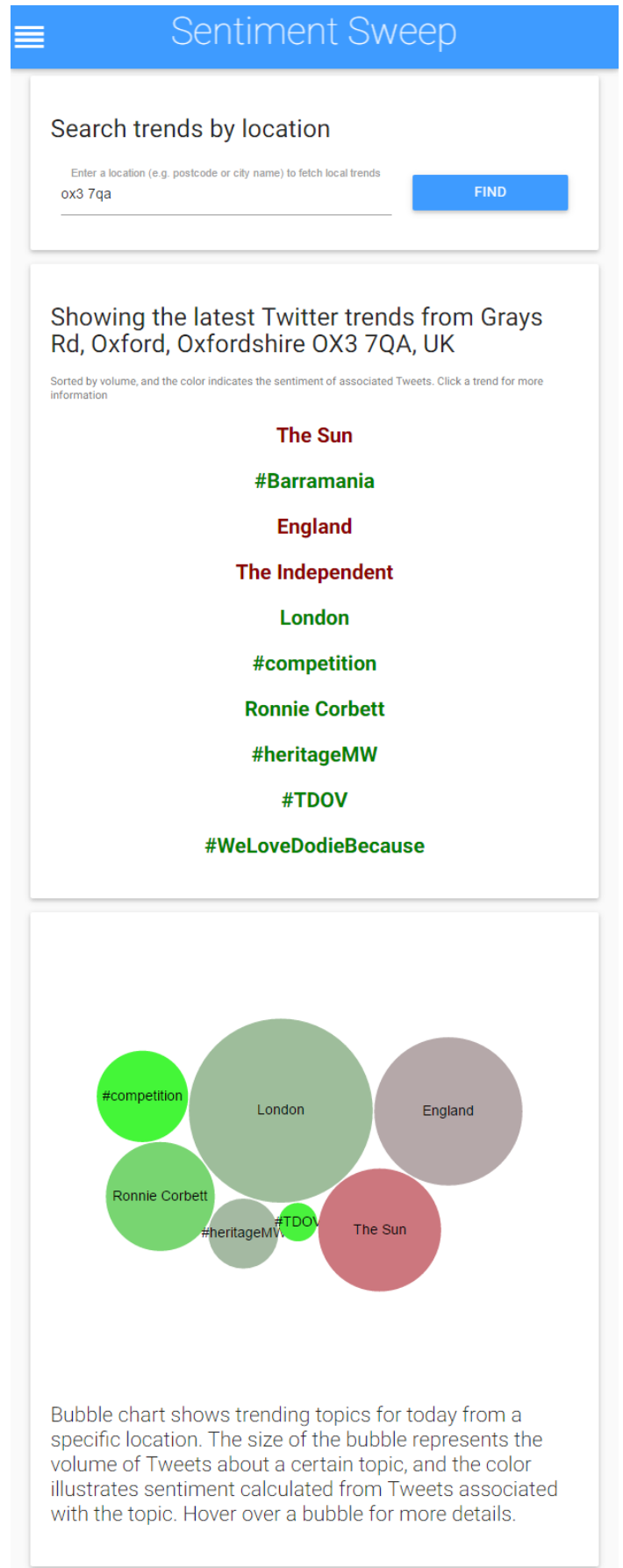


Fig 29 Screen shot of trending 2

Data flow behind the trending page

The following flow chart shows the flow of data in the backend of the trending page.

Because there is quite a lot of data requests done, where possible these are all asynchronous, and everything is initiated with an AJAX call, which means the page can render while the request is still taking place. A loading spinner is displayed while the system waits for data to be returned.

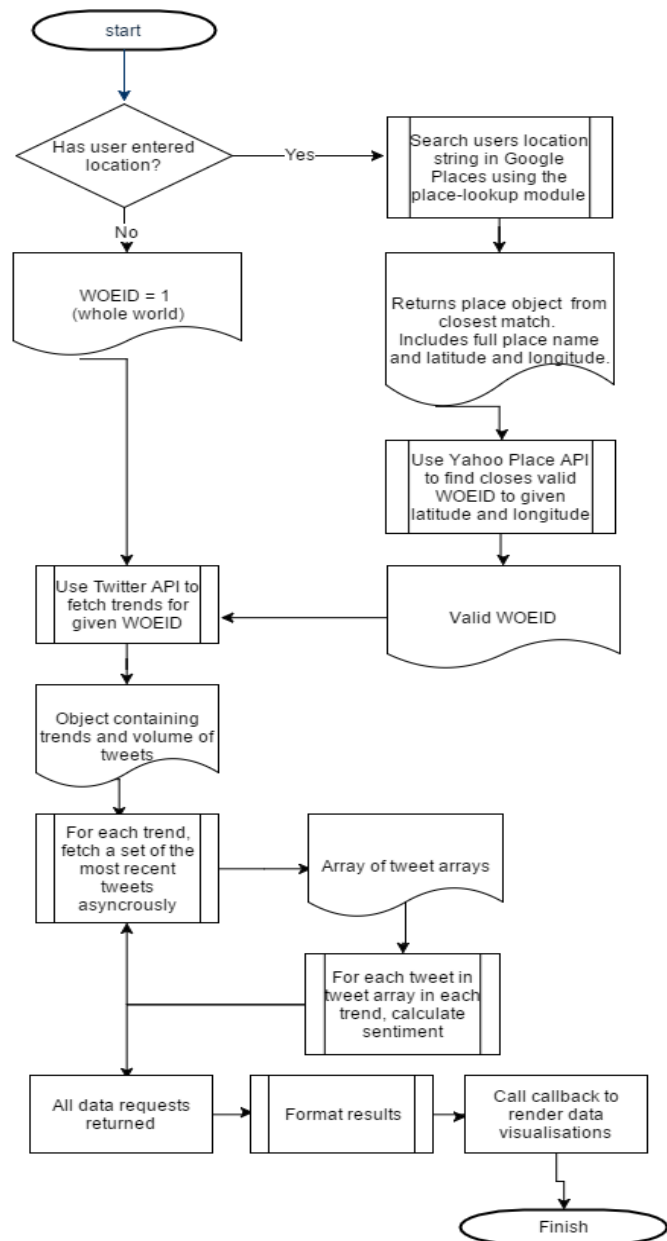


Fig 30 Data flow diagram for trending page

The following code snippet initiates the AJAX request from the client side.

```

$(document).ready ->
  urlPage = window.location.href. split('/').pop() .toLowerCase()
  urlEnd = if urlPage != 'trending' then urlPage else ''
  $.post('/api/trending/'+urlEnd, {}, (results) -> renderResults results)
  
```

Once the data has been returned, the `renderResults(results)` function is called. This checks the format of the results, hides the spinners and initiates the function to draw the bubble chart and trend list.

Rendering the bubble chart with D3

The following CoffeeScript code snippet draws the bubble chart, and binds it to the data.

```
drawBubbleChart = (results) ->
  diameter = 450
  format = d3.format(',d')

  scaleColors = ["#C80000", "#EB3443", "#B1B1B1", "#42F735", "#4AF43E"]
  color = d3.scale.linear()
    .domain([-0.8, -0.2, 0, 0.2, 0.8])
    .range(scaleColors)

  bubble = d3.layout.pack().sort(null).size([diameter, diameter]).padding(1.5)
  svg = d3.select('#bubble-trends')
    .append('svg')
    .attr('width', diameter)
    .attr('height', diameter)
    .attr('class', 'bubble')

  classes = (root) ->
    classes = []
    recurse = (name, node) ->
      if node.children
        node.children.forEach (child) ->
          recurse node.name, child
      else
        classes.push
          packageName: name
          className: node.name
          value: node.size
          col: node.col
    recurse null, root
    { children: classes }

  root = { name: 'trending', children: [
    { name: 'positive', children: [] }
    { name: 'negative', children: [] }
  ] }

  for trend in results
    if trend.sentiment > 0
      root.children[0].children.push {
        name: trend.topic, size: trend.volume, col: trend.sentiment }
    else if trend.sentiment < 0
      root.children[1].children.push {
        name: trend.topic, size: trend.volume, col: trend.sentiment }

  node = svg.selectAll('.node')
    .data(bubble.nodes(classes(root)))
    .filter((d) -> !d.children)
    .enter()
    .append('g')
    .attr('class', 'node')
    .attr('transform', (d) -> 'translate(' + d.x + ', ' + d.y + ')')

  node.append('title').text (d) -> d.className + ': ' + format(d.value)

  node.append('circle')
    .attr('r', (d) -> d.r)
    .style 'fill', (d) -> color d.col

  node.append('text')
    .attr('dy', '.3em')
    .style('text-anchor', 'middle')
    .text (d) -> d.className.substring 0, d.r / 3

  d3.select(self.frameElement).style 'height', diameter + 'px'
```

4.9.3 Comparison Chart

The aim of this page was to let the user compare different topics (e.g. comparing political leaders before an election, or sports teams, or competing brands, or locations...).

Again this page used the asynchronous tweets script as its primary backend module (since between 2 and 4 sets of tweets were fetched). The frontend was a fairly simple donut chart and word list.

The user has the choice to enter between 2 and 4 different topics to compare. The page is responsive and will adjust accordingly.

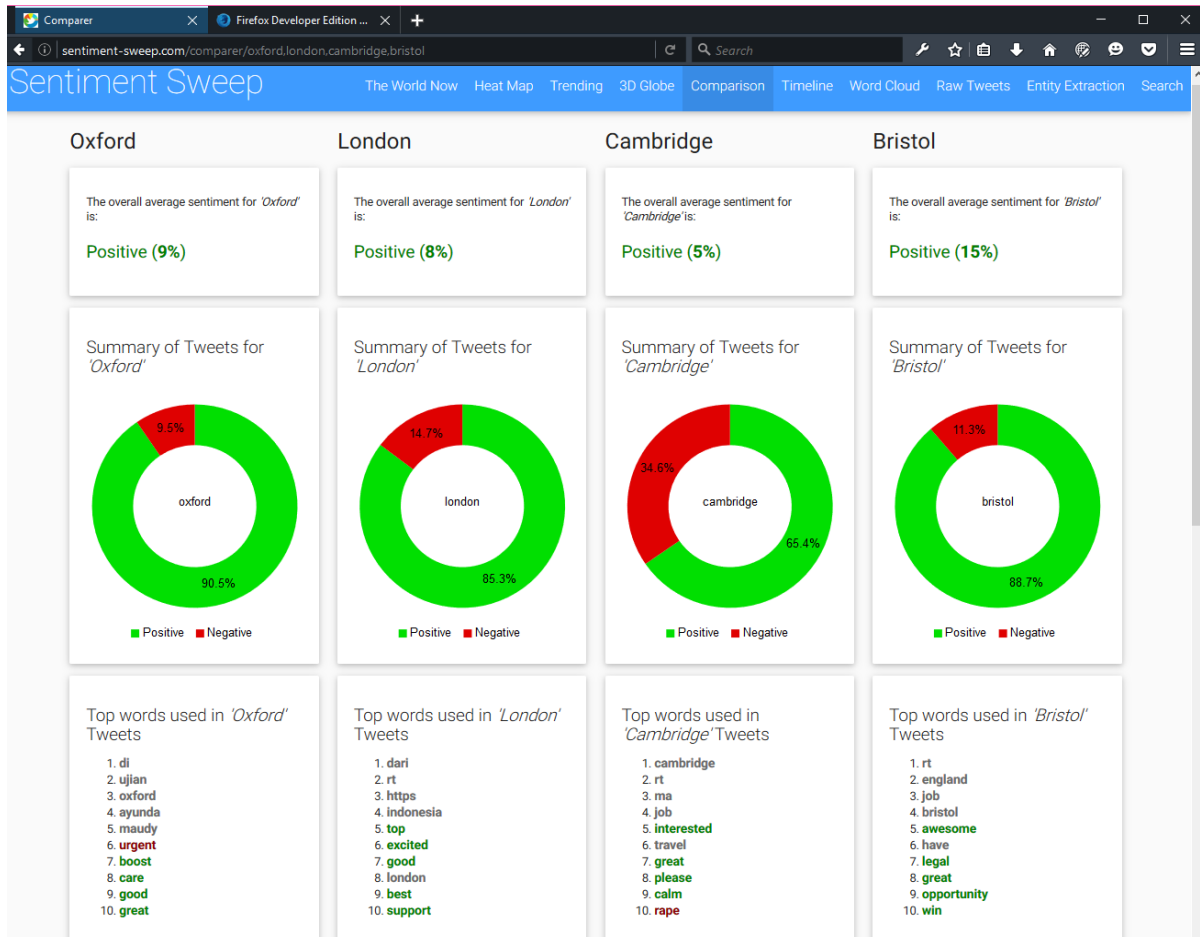


Fig 31 Screen shot for comparison

4.10 SPRINT 9 – REAL-TIME FUNCTIONALITY

4.10.1 Implementing the real-time system

During this sprint the infrastructure for real-time data updating was developed. Existing charts were updated to become real-time, and a new page (The World Now) was developed to show off the real-time efficiency.

Socket.io was the obvious choice of technologies to use, as it manages all the complexities of web sockets and data handling very nicely, and has exceptional browser support. In the methodology other options were considered.

A server-side instance of socket.io runs on the node app, and listens for an event. When it receives such an event, it notifies its subscribers (the client browsers). They then look at the data check what's new, and re-render the appropriate component of a given chart.

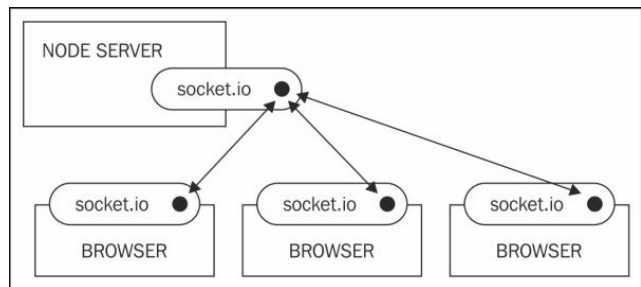


Fig 32 Diagram of node socket real-time

4.10.2 Real-time Dashboard

The real-time dashboard or “World Now” aims to give the user an overview of how the world is feeling at that point in time. Everything is live, so as soon as a new tweet comes in, it is plotted to the map, and added to the statistics.

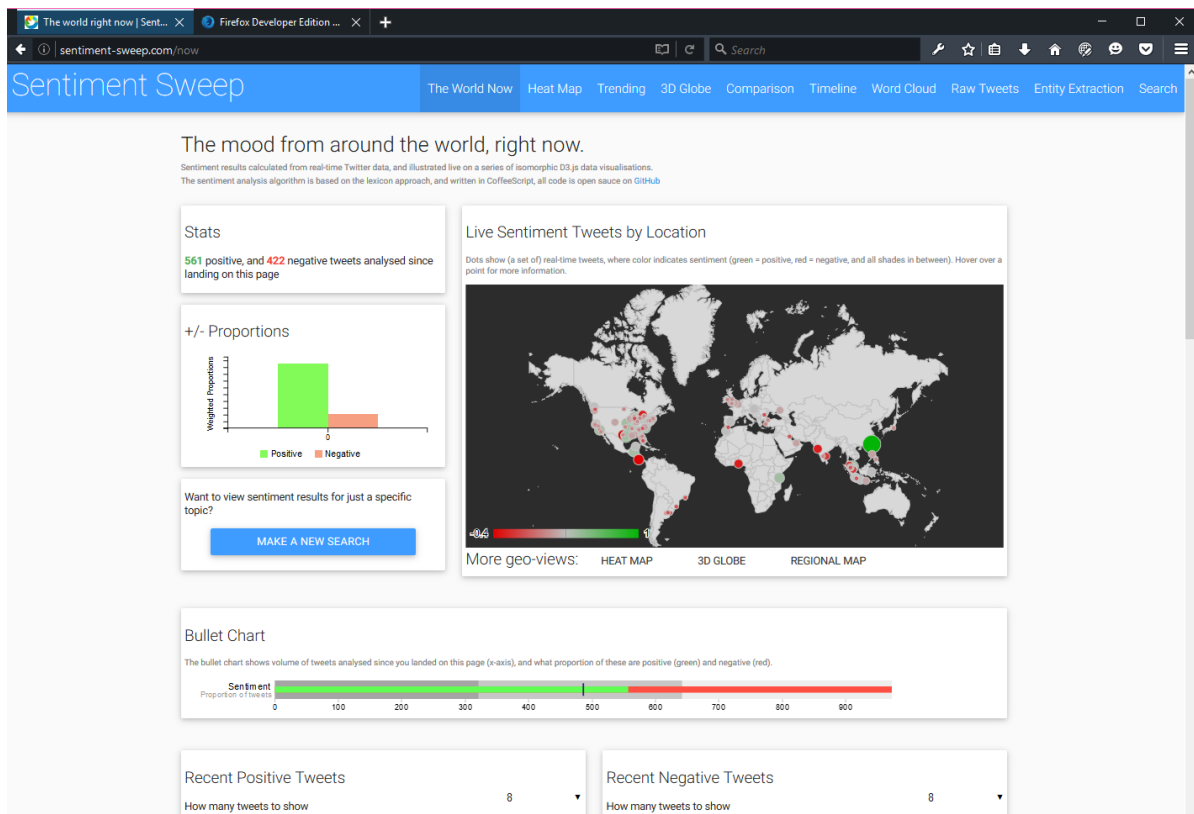


Fig 33 Screen shot of the Now page

4.11 SPRINT 10 – ADVANCED DATA VISUALISATIONS

The aim of sprint 10, was to use other API's which generate similar data, to create a series of data visualisations allowing the user to gain a more advanced insight into their topic. This included tone analysis, entity extraction and topic breakdown.

4.11.1 Tone Analysis

This section used HP Idol OnDemand, to try and determine the tone of speech conveyed in Tweets for a given topic. D3 was then used to render these results accordingly.

In the following example, the user has searched for “National Rail”. The strongest tone conveyed for this topic is anger, there is very little joy, and a small amount of fear and sadness.



Fig 34 Screen shot of the tone identification page

4.11.2 Entity Extraction

Entity extraction is the process of automatically determining document metadata from an unstructured body of text. For this section IBM Watson was utilised along with Wikipedia (as the dataset).

A Sankey chart was developed in D3 to display the results.

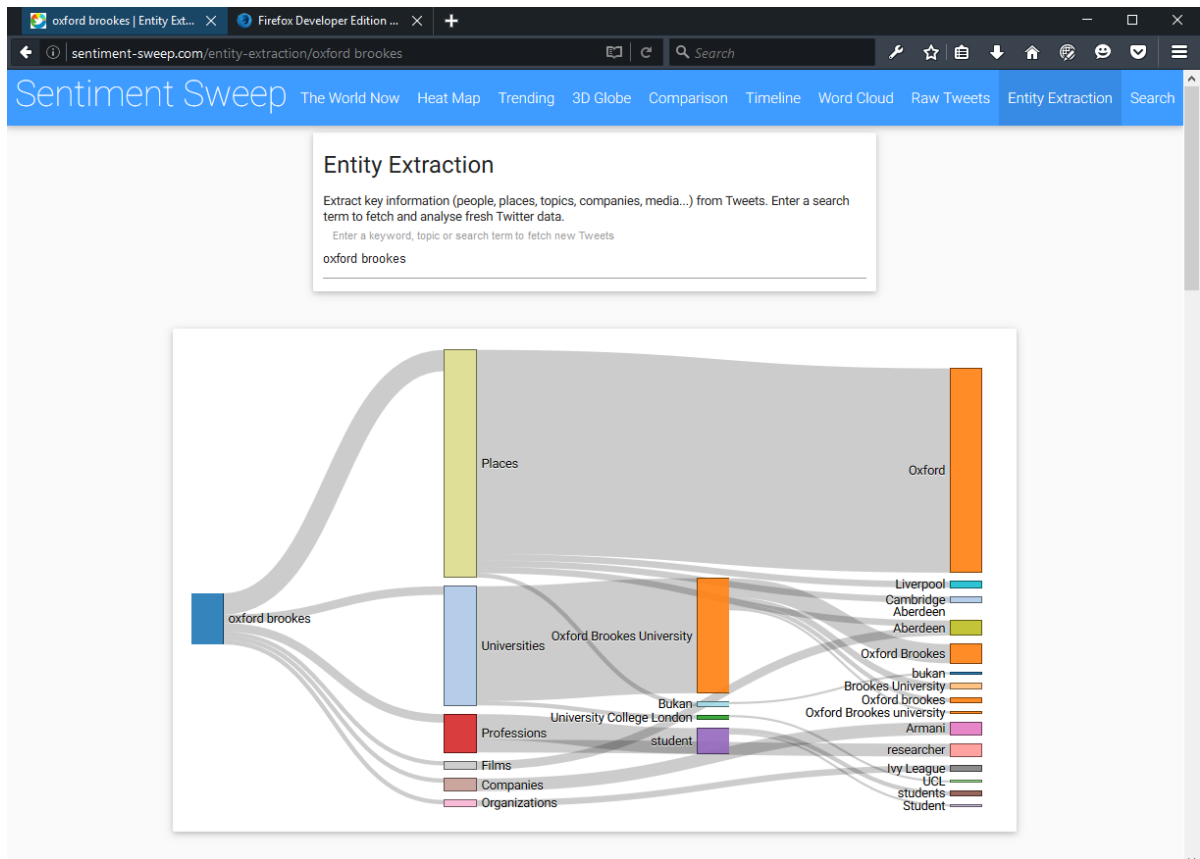
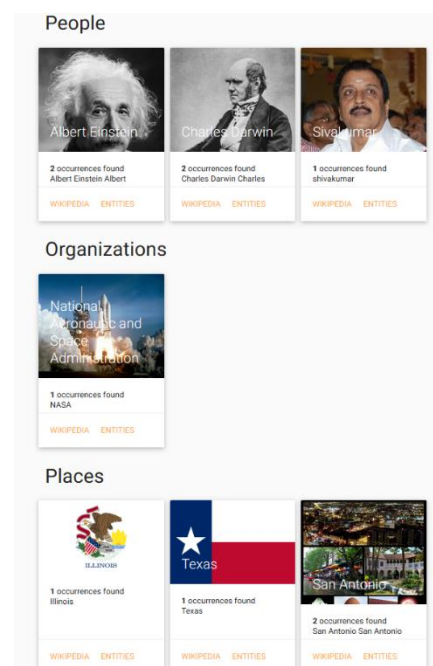


Fig 35 Screen shot of the entity extraction page

As well as the Sankey chart, a list (with graphics, pulled from Wikipedia) was used to display the results.

The snippet on the right show's a small set of entities pulled from the term "Computer Science". Other entities included languages, professions, companies, teams, films etc...



4.12 SPRINT 11 – HOME PAGE AND SEARCH PAGE

Sprint 11 was the final development sprint, it was dedicated to developing the homepage, search page and polishing of the site structure (navigation bar, footer, about page etc....).

4.12.1 The Home Page

The homepage aimed to quickly inform the user of what the application is and can be used for, as well as pointing them in the direction for getting started (either making a search, or scrolling down and clicking a data visualisation link). It is essential that the homepage is clear, but also interesting enough to engage the user. From a more practical point of view, it must also load quickly and run smoothly.

The home page is in two parts, each filling 100% of the screen, and connected by a parallax scrolling animation.

The upper section of the homepage

The first section, gives a quick introduction to what Sentiment Sweep is (with a read more button, for more detail), and also has the two entry routes – make a search, or scroll down.

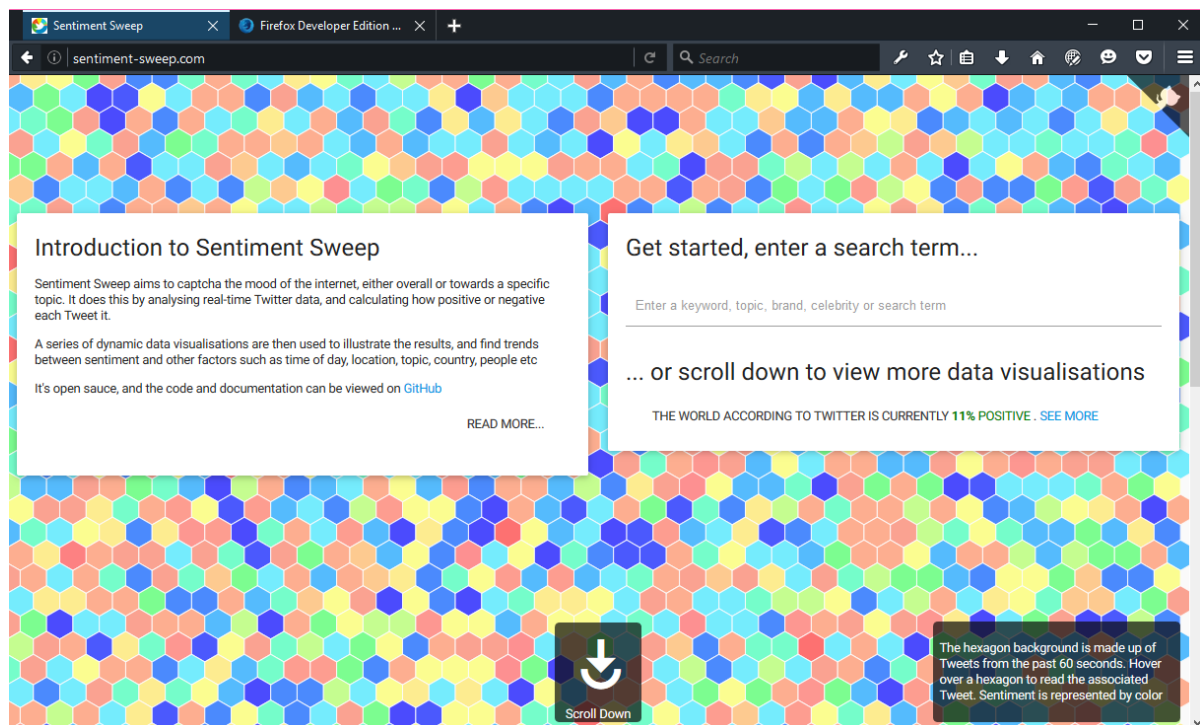
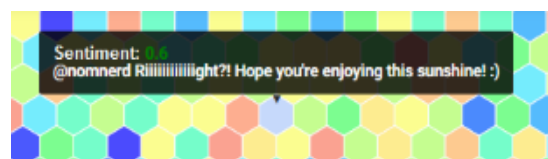


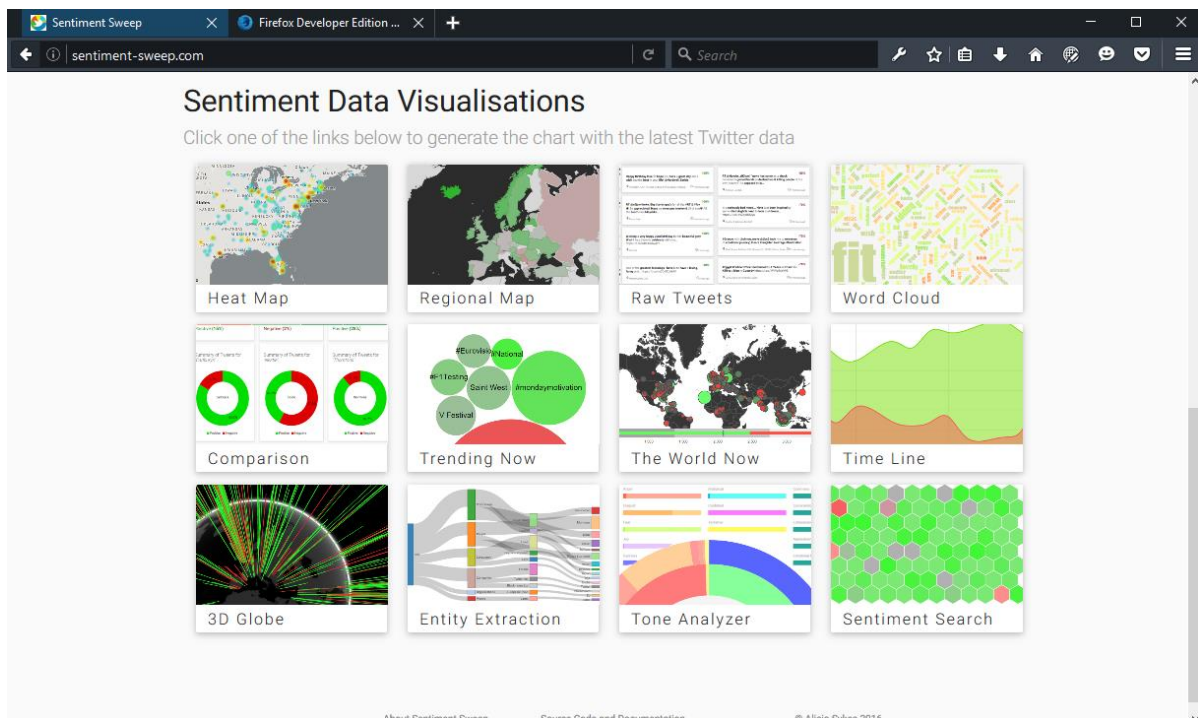
Fig 36 Screen shot of the home page

The background is made up of a series of dynamic hexagons, each represents a tweet from the past 60 seconds, the hexagons are constantly updating, as new tweets come in, in real-time. The colour of the hexagon shows the sentiment of the tweet. The user can hover over a hexagon to read a tweet. This is all explained the user briefly in the bottom right corner.



The lower section of the homepage

Once the user either scrolls down, or taps the arrow, they are directed to this screen. This page aims to give the user quick access to every main page on the application.



The twelve tiles with thumbnails represents the twelve data visualisations. Each tile is a hyperlink/ button that the user can click/ tap to be directed to the relevant page. The user can scroll back up at any time to get back to the upper section of the homepage.

Fig 37 Screen shot of second part of home page

Using traditional HTML, this page would have required a lot of mark-up code and repetition (a div, image, text and hyperlink [at least] for each tile!). However, since Jade was used as the primary templating engine, this whole section was written (using a for loop) with the following code:

```
.section#part-2
  .container
    h2 Sentiment Data Visualisations
    p.flow-text.grey-text.darken-4 Click one of the links below to generate the chart
    .row
      each page in pages
        .col.s10.offset-s1.m4.l3
          a(href='#{page.page}')
            .card.home-card
              .card-image.waves-effect.waves-block.waves-light
                img(src='/images/thumbnails/thumb_#{page.index}.png')
              .card-content
                span.card-title.grey-text.text-darken-4= page.title
```

In the layout.jade file (which this inherits from), there is a JSON object with reference to each of the twelve visualisations, that is used numerously throughout the application.

If the user instead, makes a search in the upper section, they will be redirected to the search results page, the next section outlines the key functionality of this page.

4.12.2 The Search Page

This page allows the user to make a search, for any topic and view a summary of sentiment results and links for further results. The user is directed to this page either from the homepage, by clicking a keyword in any of the data visualisations/ charts or by clicking a link directly to the search page.

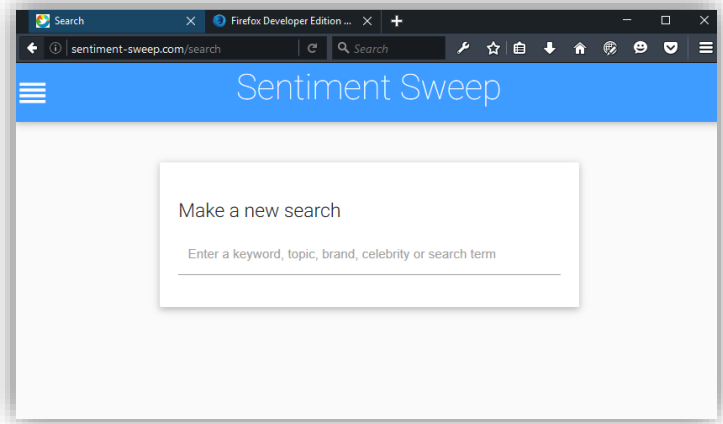
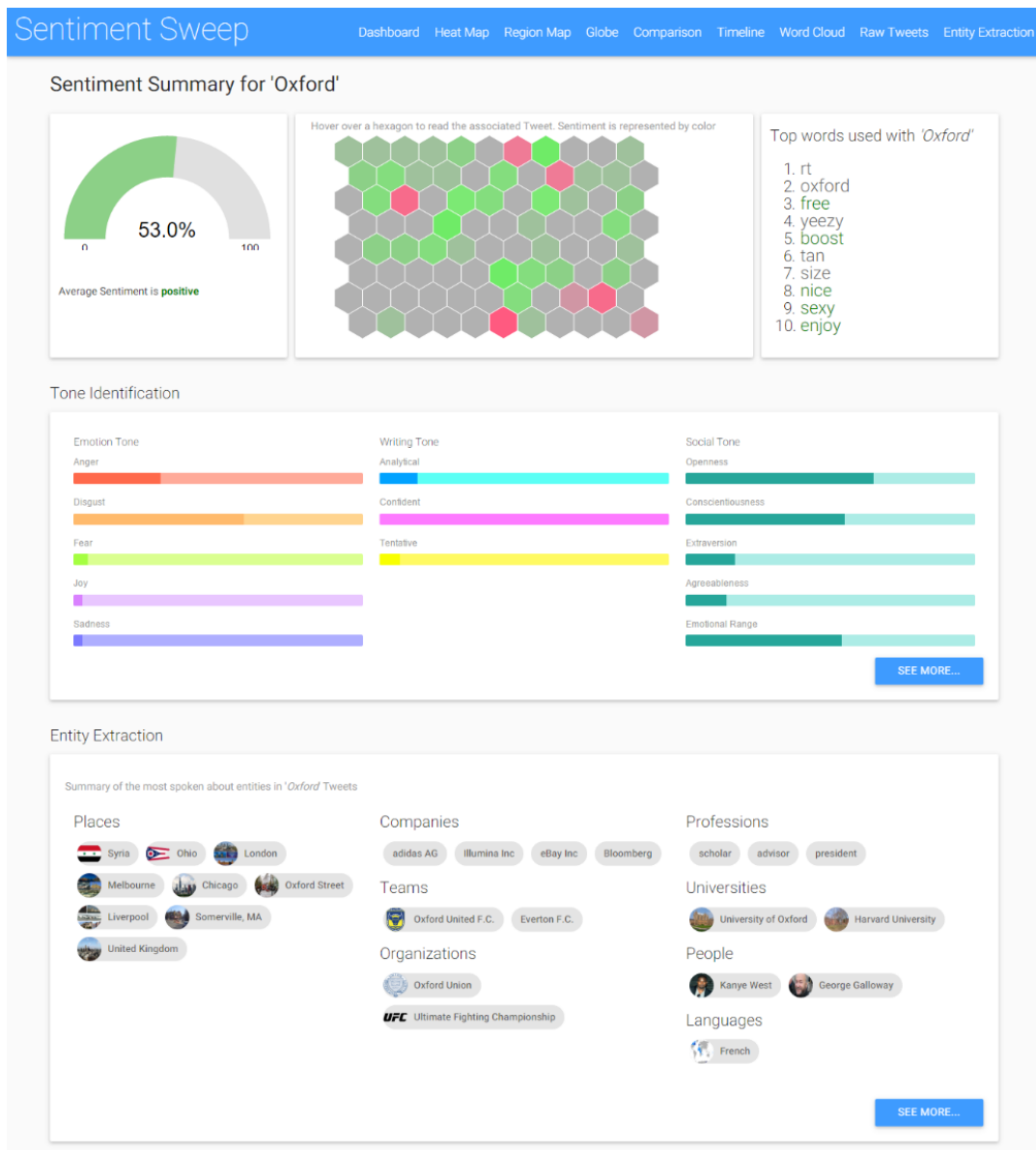


Fig 38 Screen shot of the search page

The following pages show screen shots. There was too much content to fit on a single page, so it has been broken down into several images across the next three pages.

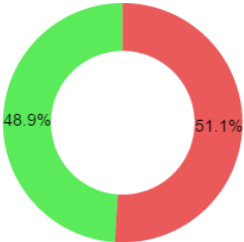
Screen shot – part 1



Screenshot – part 2

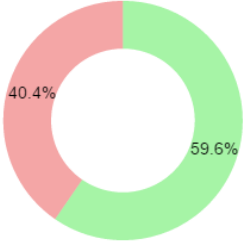
Compare 'brussles' with other topics

'brussles' Average



48.9% Positive 51.1% Negative

Overall Average for Today



40.4% Positive 59.6% Negative

Compare with more Twitter results

Enter up to four brand names, retailers, businesses, places, celebrities or objects that you'd like to compare peoples opinions on.

First Search Term
brussles

Second Search Term

[Add more search queries](#)

[GET RESULTS!](#)

Top Positive Tweets

- @A_Human_Crisis** there are **terrorists** between them , you **took** the **chance** ,#brussles #paris are you **prepared** to take more **chances** ? **50%**

📍 23:18
- RT @NetworksManager:** This is **awesome!!** #Pig's heads found on **fence** at #muslim #Morocco ambassador's #Paris home. #islamispence #Brussles ht... **40%**

📍 England, United Kingdom 17:1
- RT @NetworksManager:** This is **awesome!!** #Pig's heads found on **fence** at #muslim #Morocco ambassador's #Paris home. #islamispence #Brussles ht... **40%**

📍 16:50
- RT @NetworksManager:** This is **awesome!!** #Pig's heads found on **fence** at #muslim #Morocco ambassador's #Paris home. #islamispence #Brussles ht... **40%**

📍 Somewhere In Time 16:49
- RT @NetworksManager:** This is **awesome!!** #Pig's heads found on **fence** at #muslim #Morocco ambassador's #Paris home. #islamispence #Brussles ht... **40%**

📍 16:49

Top Negative Results

- @politico** this is his **problem** he **speaks** without **thinking** he's an **idiot!** He was the **first dumbass** to **speak** live on @CNN regarding #Brussles **-80%**

📍 17:47
- Jehlumpostnews** #PMMODI IN #BRUSSELS PAYS TRIBUTE TO TERROR VICTIMS VISITS BLAST SIGHT <https://t.co/ET6uvmlP9L> **-60%**

📍 3:59
- RT @andrewschulz:** Real talk, **Homeland predicts the future.** Predicted the **terror attacks** in Paris and **Brussles** AND said US would try and rep... **-40%**

📍 15:24
- RT @andrewschulz:** Real talk, **Homeland predicts the future.** Predicted the **terror attacks** in Paris and **Brussles** AND said US would try and rep... **-40%**

📍 21:19
- RT @andrewschulz:** Real talk, **Homeland predicts the future.** Predicted the **terror attacks** in Paris and **Brussles** AND said US would try and rep... **-40%**

📍 13:4

VIEW MORE TWEETS

Screenshot – part 3

Click on a link below to see the 'brussles' results data in another format



Heat Map



Regional Map



Raw Tweets



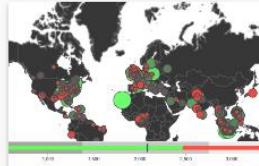
Word Cloud



Comparison



Trending Now



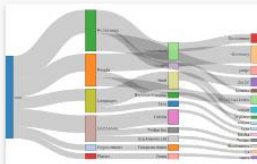
The World Now



Time Line



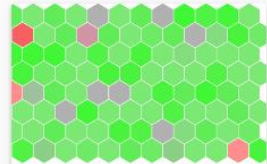
3D Globe



Entity Extraction



Tone Analyzer



Sentiment Search

Make a new search

Enter a keyword, topic, brand, celebrity or search term

Components of the Search Page

Sentiment Summary

The first section of the search results page has a dial, hexagon shape and a list of keywords. The aim of this is to give the user a super quick overview of what the overall sentiment and distribution of sentiment is.

The dial quickly shows average – a single number between -100 and +100.

However, some topics may be very controversial, where a lot of people hold strong positive opinions and others hold strong negative opinions (e.g. a politician). The purpose of the hexagon area is to show sentiment across a set of tweets. Like the homepage, each hexagon represents a tweet, colour indicates sentiment and the hexagons can be hovered over to find out more.

The words list shows the top words used in tweets relating to the searched topic. Positive words are in green, negative in red and the rest in grey. Words are clickable.

Tone Identification Summary

An AJAX request is made to the tone identification API route, and the results are displayed in a series of mini bar charts, illustrating tones such as anger and joy. There is a see more button which takes the user to the main tone identification page, which shows much more detail.

Entity Extraction Summary

This is a quick summary of the key entities extracted from the search results, it only shows the six top categories. Again there is a see more button, taking the user to the entity extraction page where they can see more details.

Comparison Summary

A quick summary of how the search results compare to the average sentiment on twitter over the past 60 minutes. There is also the option for the user to enter up to four search terms to compare sentiment with (so they can compare different baseball teams for example). This redirects the user to the comparison page.

Raw Tweets Summary

This gives a small snippet of the top most positive and most negative tweets from the set of tweets used in the search results. Keywords are highlighted in bold to make it easier for the user to skim read (and they are clickable). The time/date, location and sentiment of each tweet is also displayed. There again is a see more button, which shows the user a larger set of relevant positive and negative tweets.

Links to further data visualisations

The final section provides a series of link to the data visualisations on the rest of the site, but with accustom hyperlink for the users search term. For example the heat map will show only results relevant to what the user searched for, as will the timeline and all other charts.

4.13 SPRINT 12 – FINISHING, UX TESTING AND LAUNCHING

4.13.1 The Icon Graphic

The icon to the below is the small logo and favicon used. It show's a Twitter-bird (because the data source is Twitter), and a green-red background (because most of the application categorised data as green or red).



Fig 39 Logo

4.13.2 Launching

The app was packaged and published as described in the methodology. The gulp build command was used to create the compiled version of the code. NGINX was used as a DNS proxy on an Apache server. Forever was used to start the final node app, and keep it running, it is initiated and monitored vis SSH. The files were uploaded via FTP onto a VPS. And the following domain points to the node app which is running on port 3000 of 109.73.175.184 (<http://109.73.175.184:3000/>).

<http://sentiment-sweep.com/>

4.13.3 UX Testing

At various points during this spring, a set of users tested out the application. They were asked to fill in a survey that covered aspects such as ease of use, clarity of results, expected accuracy etc. They rated each criterion (quantitatively) between 0 and 5 on each of these points, and if the score was under 5 they also briefly gave a sentence or two on how it could be improved (qualitatively). Once these improvements were implemented the user completed the survey again. This process was repeated three times during this final sprint.

The full survey can be viewed in the appendix.
Below is a summary of the quantitative results for each iteration.

4.13.4 User Experience Survey Iteration 1 (initial increment)

User Experience Review 1								
Page/ Section	Criteria	Joe	George	Fiona	Jen	Rosie	Ollie	Average
About User	Age Range	> 18	18 - 25	18 - 25	25 - 65	18 - 25	18 - 25	-
	Technical Ability	4	5	3	2	3	5	-
Search Page	Clarity of Results	5	4	4	5	5	5	4.67
	Usefulness of Results	5	5	5	5	4	5	4.83
	Accuracy of Results	4	5	3	4	4	5	4.17
	Ease of interpretation	3	5	5	3	5	5	4.33
Tone Identification	Clarity of Results	5	5	5	5	5	5	5.00
	Usefulness of Results	5	5	4	5	5	5	4.83
	Accuracy of Results	5	3	2	5	5	3	3.83
	Ease of interpretation	5	5	5	5	5	5	5.00
Entity Extraction	Clarity of Results	5	5	4	5	5	5	4.83
	Usefulness of Results	4	5	5	4	5	4	4.50
	Accuracy of Results	5	5	5	5	5	5	5.00
	Ease of interpretation	5	5	5	5	5	5	5.00
Topic Comparison	Clarity of Results	5	5	4	5	5	5	4.83
	Usefulness of Results	5	5	5	5	5	5	5.00
	Accuracy of Results	5	4	5	5	5	4	4.67
	Ease of interpretation	5	5	5	5	5	5	5.00
Raw Tweets	Usefulness of Results	5	5	3	5	5	4	4.50
	Ease of interpretation	5	5	5	5	5	5	5.00
Heat Map	Ease of interpretation	4	5	4	4	4	5	4.33
	Clarity of Results	4	5	4	4	5	5	4.50
Regional Map	Ease of interpretation	5	5	5	5	5	5	5.00
	Clarity of Results	5	5	5	5	5	5	5.00
Word Cloud/ Scatter Plot	Clarity of Results	4	4	5	4	4	5	4.33
	Usefulness of Results	4	4	3	5	5	5	4.33
	Accuracy of Results	5	5	5	5	5	5	5.00
	Ease of interpretation	5	5	5	5	5	5	5.00
3D Globe	Overall Rating	5	5	5	5	4	5	4.83
Trending	Overall Rating	5	5	5	5	5	5	5.00
Time Line	Overall Rating	2	3	2	2	3	4	2.67
Home Page	Clarity of app definition	5	5	4	5	3	5	4.50
	Clarity of initial routes	4	5	4	4	5	5	4.50
Overall	Average	4.61	4.74	4.35	4.65	4.71	4.81	4.65

Table 14 User experience survey 1

4.13.5 User Experience Survey Iteration 2

User Experience Review 2								
Page/ Section	Criteria	Joe	George	Fiona	Jen	Rosie	Ollie	Average
About User	Age Range	> 18	18 - 25	18 - 25	25 - 65	18 - 25	18 - 25	-
	Technical Ability	4	5	3	2	3	5	-
Search Page	Clarity of Results	5	5	5	5	5	5	5.00
	Usefulness of Results	5	5	5	5	4	5	4.83
	Accuracy of Results	4	5	4	5	5	5	4.67
	Ease of interpretation	3	5	5	3	5	5	4.33
Tone Identification	Clarity of Results	5	5	5	5	5	5	5.00
	Usefulness of Results	5	5	4	5	5	5	4.83
	Accuracy of Results	5	5	3	5	5	4	4.50
	Ease of interpretation	5	5	5	5	5	5	5.00
Entity Extraction	Clarity of Results	5	5	5	5	5	5	5.00
	Usefulness of Results	4	5	5	4	5	5	4.67
	Accuracy of Results	5	5	5	5	5	5	5.00
	Ease of interpretation	5	5	5	5	5	5	5.00
Topic Comparison	Clarity of Results	5	5	5	5	5	5	5.00
	Usefulness of Results	5	5	5	5	5	5	5.00
	Accuracy of Results	5	5	5	5	5	4	4.83
	Ease of interpretation	5	5	5	5	5	5	5.00
Raw Tweets	Usefulness of Results	5	5	3	5	5	5	4.67
	Ease of interpretation	5	5	5	5	5	5	5.00
Heat Map	Ease of interpretation	5	5	5	5	5	5	5.00
	Clarity of Results	5	5	5	5	5	5	5.00
Regional Map	Ease of interpretation	5	5	5	5	5	5	5.00
	Clarity of Results	5	5	5	5	5	5	5.00
Word Cloud/ Scatter Plot	Clarity of Results	4	4	5	4	4	5	4.33
	Usefulness of Results	4	4	3	5	5	5	4.33
	Accuracy of Results	5	5	5	5	5	5	5.00
	Ease of interpretation	5	5	5	5	5	5	5.00
3D Globe	Overall Rating	5	5	5	5	4	5	4.83
Trending	Overall Rating	5	5	5	5	5	5	5.00
Time Line	Overall Rating	2	3	2	2	3	4	2.67
Home Page	Clarity of app definition	5	5	4	5	3	5	4.50
	Clarity of initial routes	4	5	4	4	5	5	4.50
Overall	Average	4.68	4.87	4.58	4.74	4.77	4.90	4.76

Table 15 User Experience survey 2

4.13.6 User Experience Survey Iteration 3 (final solution)

User Experience Review 3								
Page/ Section	Criteria	Joe	George	Fiona	Jen	Rosie	Ollie	Average
About User	Age Range	> 18	18 - 25	18 - 25	25 - 65	18 - 25	18 - 25	-
	Technical Ability	4	5	3	2	3	5	-
Search Page	Clarity of Results	5	5	5	5	5	5	5.00
	Usefulness of Results	5	5	5	5	5	5	5.00
	Accuracy of Results	5	5	5	5	5	5	5.00
	Ease of interpretation	5	5	5	5	5	5	5.00
Tone Identification	Clarity of Results	5	5	5	5	5	5	5.00
	Usefulness of Results	5	5	5	5	5	5	5.00
	Accuracy of Results	5	5	4	5	5	5	4.83
	Ease of interpretation	5	5	5	5	5	5	5.00
Entity Extraction	Clarity of Results	5	5	5	5	5	5	5.00
	Usefulness of Results	5	5	5	5	5	5	5.00
	Accuracy of Results	5	5	5	5	5	5	5.00
	Ease of interpretation	5	5	5	5	5	5	5.00
Topic Comparison	Clarity of Results	5	5	5	5	5	5	5.00
	Usefulness of Results	5	5	5	5	5	5	5.00
	Accuracy of Results	5	5	5	5	5	4	4.83
	Ease of interpretation	5	5	5	5	5	5	5.00
Raw Tweets	Usefulness of Results	5	5	5	5	5	5	5.00
	Ease of interpretation	5	5	5	5	5	5	5.00
Heat Map	Ease of interpretation	5	5	5	5	5	5	5.00
	Clarity of Results	5	5	5	5	5	5	5.00
Regional Map	Ease of interpretation	5	5	5	5	5	5	5.00
	Clarity of Results	5	5	5	5	5	5	5.00
Word Cloud/ Scatter Plot	Clarity of Results	5	5	5	5	5	5	5.00
	Usefulness of Results	5	5	5	5	5	5	5.00
	Accuracy of Results	5	5	5	5	5	5	5.00
	Ease of interpretation	5	5	5	5	5	5	5.00
3D Globe	Overall Rating	5	5	5	5	5	5	5.00
Trending	Overall Rating	5	5	5	5	5	5	5.00
Time Line	Overall Rating	5	5	5	5	5	5	5.00
Home Page	Clarity of app definition	5	5	5	5	5	5	5.00
	Clarity of initial routes	5	5	5	5	5	5	5.00
Overall	Average	5.00	5.00	4.97	5.00	5.00	4.97	4.99

Table 16 User experience survey 3

5 EVALUATION

This chapter evaluates the final solution that was implemented, and how well it solves the problem outlined in the introduction. The aim of the evaluation is to allow

The system will be evaluated based on the following criteria:

- **Meeting requirements** – The degree of compliance with the user stories and acceptance criteria specified in the methodology
- **User experience** – How useful, clear and enjoyable the users found using the application
- **Time management and scheduling** – How closely the proposed schedule was followed
- **Technical testing**– Summary of unit test results
- **Code quality** - Quality and style of code and project structure
- **Self-evaluation** - Discussion of how well I thought the project went compared to the plan

5.1 EVALUATION OF REQUIREMENTS MET

This section reviews each of the system and functional requirements, that were specified in the form of user stories in the methodology. The table shows whether or not the system meets each point. If it does not, and explanation is given. Only the user stories for the main application are listed. To read the acceptance criteria for each story, see Appendix 1.

Sprint	Story ID	User Story	Complete	Notes
Sprint 0	TSV-T075	As a development environment I should compile CoffeeScript into JavaScript	✓	
	TSV-T076	As a development environment I should compile LESS/ SASS into CSS	✓	
	TSV-T077	As a development environment I should check HTML/ Jade for missing entities	✓	
	TSV-T078	As a development environment I should prepare graphics from source to their web form	✓	
	TSV-T079	As a development environment I should remove obsolete files	✓	
	TSV-T080	As a development environment I should bundle browser scripts	✓	
	TSV-T081	As a development environment I should check code (where possible) for errors /bad style	✓	
	TSV-T082	As a development environment I should watch for changes in source files and then recompile the appropriate files	✓	
	TSV-T083	As a development environment I should keep multiple development browsers across various devices in sync for seamless compatibility testing	✓	<i>Only happens when the default gulp process or browser-sync task is run</i>

	TSV-T084	As a development environment I should run unit tests and output results	✓	<i>Runs automatically when a test file changes, or when the test command is run</i>
	TSV-U085	As a test environment I should run unit tests	✓	
	TSV-U086	As a test environment I should have integration tests	✓	
Sprint 1	TSV-A001	As a user I should be able to view the solution in my browser simply by visiting a URL	✓	<i>http://sentiment-sweep.com</i>
	TSV-A002	As a user I should have fast loading times or be displayed with a loading graphic	✓	<i>Usually, yes - but depends on user factors (such as internet speeds and browser) as well</i>
	TSV-A003	As a user I should be able to easily switch between different parts of the application	✓	<i>Using navigation bar</i>
	TSV-A004	As a user I should find the user interface clear and simple to look at	✓	
	TSV-U087	As a test environment I should be able to stub data as to not make external data requests	✗	<i>Wasn't necessary for the main application, since all modules had data stubbed tests</i>
	TSV-U088	As a test environment I should use assertions for the unit tests	✓	
	TSV-U089	As a test environment I should show coverage test results	✓	
	TSV-U090	As a test environment I should check dependencies are up-to-date	✓	
	TSV-U091	As a test environment I should check code quality with automated code reviews	✓	
	TSV-U092	As a test environment I should do headless browser testing and HTTP service testing	✗	<i>HTTP service testing was implemented, but it wasn't necessary to use it to test all parts of the application</i>

Sprint 2	TSV-E021	As a system I should be able to fetch a specified number of Tweets from around a given location from the Twitter API	✓	
	TSV-E022	As a system I should be able to fetch a specified number of Tweets talking about a specific topic, hashtag or keyword from the Twitter API	✓	
	TSV-E023	As a system I should have reasonable response times in fetching Tweets from the Twitter API	✓	
	TSV-E024	As a system I should respond with a suitable error if there is a problem fetching Tweets from the Twitter API	✓	
	TSV-E025	As a system I should be able to create a series of Tweet objects ready for the next stage	✓	
	TSV-V093	As a developer I should be able to get a continuous stream of chunked tweets	✓	
	TSV-V094	As a developer, the streamed tweets should be formatted and safe	✓	
Sprint 3	TSV-X097	As A developer I can get A valid latitude and longitude for any fuzzy place name	✓	
	TSV-X098	As a developer I can get the country code in multiple forms from latitude and longitude	✓	
	TSV-X099	As a developer I can get location of geo-tagged tweets from their ID's	✓	
	TSV-Z100	As a system I am super-efficient (and totally awesome)	✓	
	TSV-E025	As a system I should be able to create a series of Tweet objects ready for the next stage	✓	
	TSV-V094	As a developer, the streamed tweets should be formatted and safe	✓	
Sprint 4	TSV-D015	As a system I should be able to identify weather a string is positive, neutral or negative overall	✓	
	TSV-D016	As a system I should be able to identify weather the attitude towards a certain specified topic is positive, neutral or negative	✓	
	TSV-D017	As a system I should be able to give a confidence estimate showing how likely it is that the sentiment value is accurate	✗	<i>This works if used in conjunction with another service, such as IBM Watson, but not alone.</i>
	TSV-D018	As a system I should be able to return emotion analysis results for a given string	✓	

	TSV-D019	As a system the load time for fetching results should be sufficiently fast	✓	
	TSV-D020	As a system I should give a suitable response and error code if no result is available	✗	<i>Just returns empty array, wasn't possible or necessary to generate error codes</i>
Sprint 5	TSV-F026	As a user I should be able to see the heat map bound to data to display results	✓	
	TSV-F027	As a user I should be able to see search for a keyword and location to see the heat map for just that topic	✓	
	TSV-F028	As a user I should be able to hover over specific areas of interest and see what is trending or causing that particular heat patch	✗	<i>The user can click a heat patch and read tweets, but not hover to read trends - as there was just too much data for this to be efficient</i>
	TSV-F029	As a system I should be displaying data both from the database and live from Tweets	✓	
	TSV-F030	As a system I should be able to load fast and display suitable loading graphic when loading times are more than 1 second	✓	
	TSV-G031	As a system I should be able to write to the database asynchronously	✓	
	TSV-G032	As a system I should be able to read from the database asynchronously	✓	
Sprint 6	TSV-G032	As a system I should be able to read from the database asynchronously	✓	
	TSV-G033	As a system I should be able to query the database for just relevant Tweets	✓	
	TSV-G034	As a system I should delete the oldest entries when there reaches a certain number of records	✓	
	TSV-G035	As a system I should be able to check and make safe data before inserting	✓	
	TSV-G036	As a system I should use both the cached and live data in conjunction or depending on user selection	✓	
	TSV-Z100	As a system I am super-efficient (and totally awesome)	✓	
Sprint 7	TSV-B005	As a user I should be able to view the map	✓	

	TSV-B006	As a user I should be able to pan (move the map with pointing device) and zoom (with both control buttons and hardware such as mouse wheel)	✓	
	TSV-B007	As a user I should be able to search for a specific location	✓	
	TSV-B008	As a user I should find the map clear and neat to look at.	✓	
	TSV-C009	As a user I should be able to see the heat map over the top of the standard map	✓	
	TSV-C010	As a user I should be able to toggle the display of the heat map to show and hide it	✓	
	TSV-C011	As a user I should be able to adjust the opacity of the heat map with a simple slider	✓	
	TSV-C012	As a system I should bind data to the heat map	✓	
	TSV-C013	As a system I should be able to bind additional data to the map to update it after the initial heat map has already been rendered	✓	
	TSV-C014	As a user I should find the colors of the heat map clearly represent the data it is displaying	✓	
Sprint 8	TSV-L050	As a user I can see a word cloud after entering my search term	✓	
	TSV-L051	As a user I can see a text list of the top words used in positive and negative tweets	✓	
	TSV-M052	As a user I can view the scatter plot for a given search term/ keyword	✓	
	TSV-M053	As a user I can hover over a point to view the associated key word	✓	
	TSV-W095	As a developer I should be able to extract an array of keywords from a string	✓	
	TSV-W096	As a developer, I should be able to customise the deletion word set	✓	
	TSV-H037	As a user I should find the start page clear and concise	✓	
	TSV-H038	As a user I can make a search directly from the home page	✓	
	TSV-H039	As a user I can navigate to any other part of the application directly from the home screen	✓	
Sprint 9	TSV-J043	As A user I should be able see what is currently trending worldwide	✓	
	TSV-J044	As a user I should be able to enter a custom location to view local trends	✓	
	TSV-J045	As a user I can click a trend to find out more	✓	

	TSV-J046	As a user I can view trends visually, to get an overview of volume and sentiment	✓	
	TSV-O057	As a user I can see both positive and negative sentiment over the past 24 hours	✓	
	TSV-O058	As a user I can interact with the map and the axis	✓	
	TSV-P059	As a user I can enter between one and four search terms	✓	
	TSV-P060	As a user I can see the overall sentiment for each of my search terms	✓	
	TSV-P061	As a user I can see most commonly used words and their sentiment for each topic	✓	
	TSV-P062	As a user I can view links to the other data visualisations for each of the topics	✓	
Sprint 10	TSV-V093	As a developer I should be able to get a continuous stream of chunked tweets	✓	
	TSV-V094	As a developer, the streamed tweets should be formatted and safe	✓	
	TSV-Z100	As a system I am super-efficient (and totally awesome)	✓	
	TSV-K047	As a user I should be able to read the plain text tweets relating to a chosen topic/ word	✓	
	TSV-K048	As a user I should be able to quickly pick out the key information while reading	✓	
	TSV-K049	As a user I should see new tweets come in in real-time	✓	
Sprint 11	TSV-S067	As a user I can make a search	✓	
	TSV-S068	As a user I can get a quick overview of the average sentiment of my topic	✓	
	TSV-S069	As a user I can see which keywords are most commonly used in the twitter results	✓	
	TSV-S070	As a user I can see a summary of the tones identified	✓	
	TSV-S071	As a user I can see a summary of the entities extracted	✓	
	TSV-S072	As a user I can see how my topic compares to the rest of Twitter	✓	
	TSV-S073	As a user I can read a set of tweets relating to my topic	✓	
	TSV-S074	As a user I can view more results for my search term on each data visualisation	✓	
	TSV-S074	As a user I can view more results for my search term on each data visualisation	✓	

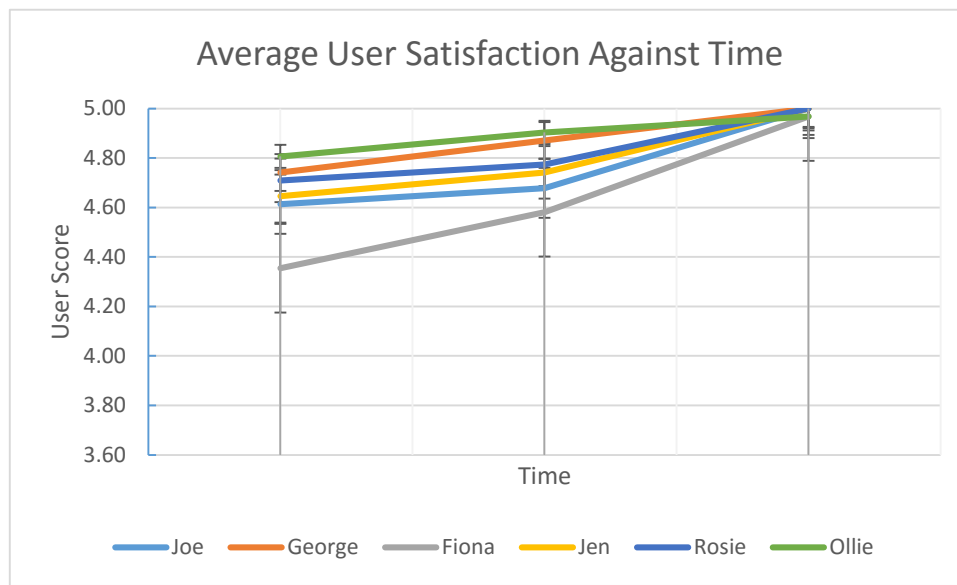
Table 17 Final evaluation of user stories

5.2 USER EXPERIENCE EVALUATION

During sprint 12, a user experience survey was conducted with a set of users, their feedback was gathered, the application was amended accordingly, and then the survey was repeated. This was repeated until the final application was close to perfect.

A summary of the results can be found in the sprint 12 section of implementation, and the full survey is included in the appendix.

The following line chart summarises the increase in user satisfaction from the results.

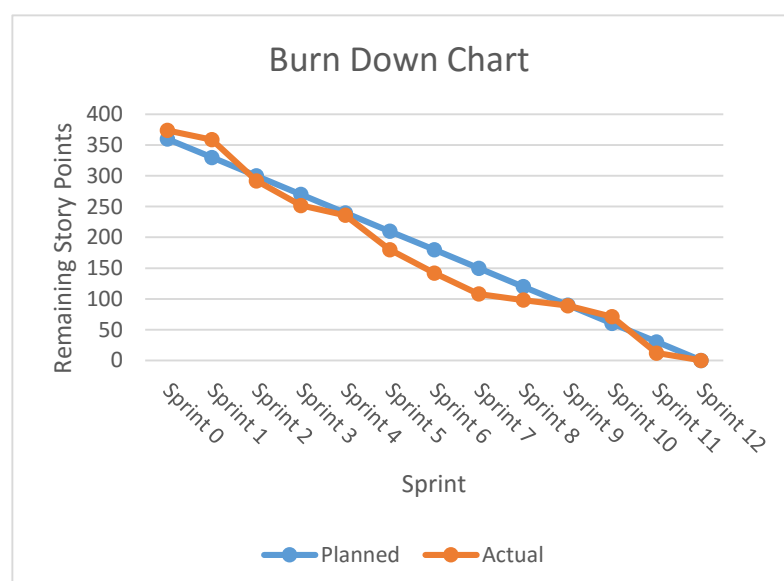


The findings from the user experience survey conclude that, although at first there were some minor usability issues, by the end of the final sprint, the average user satisfaction score was 4.99/ 5. Meaning overall users found using the application, to be a 99.8% positive experience.

5.3 TIME MANAGEMENT AND SCHEDULING

The schedule outlined in the methodology was followed very closely.

The following burndown charts shows the remaining story points (y-axis) against sprint (x-axis). When the actual (orange) is below the planned (blue) line, that indicates the project being ahead of schedule.



5.4 EVALUATION OF TECHNICAL TESTING

As the methodology explained, all appropriate aspects of the final solution should be thoroughly unit tested. This helps guard against regression bugs, and ensures a much higher quality of software.

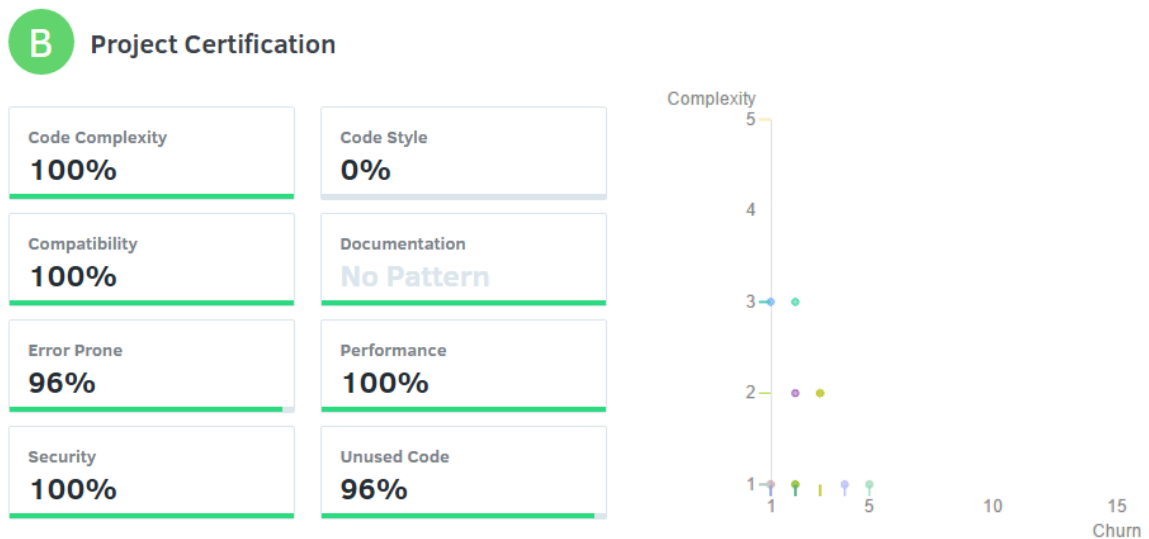
By the end of the final sprint, all unit tests were fully passing. The projects status on the continuous integration platform, also shows the apps build is passing.

A full breakdown of each aspect of the tests can be found in appendix 4.

5.5 EVALUATION OF CODE QUALITY

Another factor that helps ensure a high quality and stable final solution is code quality. Good code quality can be achieved though following standard practices and/or a language-specific style guide, getting other developers to review your code, using automated code review tools, deleting obsolete lines, functions and files and regularly refactoring.

All of the above were utilised in this project. Below is the final metrics for the automated code quality checks. (See appendix 4 for full results.)



5.6 EVALUATION OF ADDRESSING LEGAL, SOCIAL AND ETHICAL ISSUES

Throughout the research, planning, development, implementation and reviews of the final application several potential legal, social and ethical issues came up. Each was addressed accordingly, and significant research was done to ensure that the final solution does not cause any further issues.

5.6.1 Handling people's personal data

The only personal data that the application handles are tweets. In order to protect the creator of each tweet used, the following precautions were made:

- Only tweets that the author has explicitly made public will be fetched from twitter
- The user handle (their username), will be eradicated from the tweet before it is displayed
- If the tweet is geo-tagged, the location will be blurred by up to +/- 100m.
- No tweet will ever be stored in the system for more than 24 hours. (Usually less than 60 minutes.)
- Some tweets that contain particularly aggressive/offensive will be filtered by the clean data algorithm. However, this is quite primitive and many will get through.
- Users are warned in the T&C's that they use the application at their own risk

5.6.2 User experience surveys

During the research (sentiment analysis computer vs human), sprint 12 (initial UX review) and the evaluation (final user reviews). Several volunteers were asked to fill in surveys or attend an informal interview. To protect these user's identity, the following steps were taken:

- The participant was either assigned an identifier (such as user 1, user 2) or only ever addressed by their first name – their full name was never used
- For personal questions (such as age, home town...) the participant just had to select a range (e.g. 18-15 or South East England) no specifics which could be used to identify the participant's identity were published

5.6.3 Offensive content

Since data is pulled directly from Twitter, there is sometimes a chance that text which may cause offence to some users may be present. To minimise this the following steps were taken:

- A simple algorithm checks and filters out some aggressive/racist or inappropriate content
- Plans for this algorithm to be expanded further, to be more complete are underway
- In the terms and conditions the users are warned that the content on the application is pulled straight from twitter, and to use at their own will, as they would with twitter

5.6.4 Accessibility

It is a legal requirement that all computer applications addressed to the general public must meet a minimum level of accessibility, so as to not exclude some user (e.g. partially sighted using screen readers etc.). To make the application accessible the following steps were taken:

- The base application is compatible with screen readers. However, for obvious reasons some of the data visualisations will not work. This is outlined in the T&C's.
- The core colour scheme for the app is red and green, this would exclude those users with colour blindness, so the program was tested with Spectrum (a Chrome extension that colour blind users use in order to adjust the red green levels on the screen). The app works fine with the extension

Further accessibility issues were discussed in the T&C's

5.7 SELF-EVALUATION

In the introduction, the problem of having too much data available, and the proposed solution was outlined. It is fair to say that the planned application was quite ambitious as it combined many new technologies together to create a complex solution with a simple interface, very different from anything else that currently exists. It also needed to be able to cope with a huge amount of data and hence the code written needed to be efficient and stable.

In the methodology the plan was outlined, and included how everything would be developed, configured and published in order to get to the final outcome. A lot of research needed to be undertaken in order to write this project plan. User surveys, experiments and literature reviews were conducted to determine the most efficient and accurate approach to the sentiment analysis algorithm. Research and upskilling was also undertaken to select the most appropriate languages, technologies and infrastructure.

The implementation section described the progress made at the end of each of the twelve sprints. It clearly demonstrated that each of the requirements had either been met, or explained why they had not been fulfilled.

5.7.1 Points that went well

- A lot of research was undertaken before the methodology, meaning the best option could be chosen before development started, and there was less that needed to be changed mid-way through.
- The methodology was detailed enough to follow closely when it came to development
- Maximum use of available coding tools was made, this had a big impact on time and efficiency as well as the quality of the final solution
- Significant time was spent upskilling before development of the project started

5.7.2 Points to improve on next time

- It would have helped to test the application out with a panel of industry experts as well as just everyday average users
- Some resources were not available which could have made the final project significantly better, such as access to the full Twitter history API and access to a very powerful server
- The web design, (despite having a lot of time and thought put into it), is not to a very high quality and takes away from the overall experience of the final solution
- Many other features were included in the application that wasn't even mentioned in the user stories. This kind of scope-creep could have hindered the quality of the vital components of the application (although in this case it didn't, it is something to be aware of next time).
- The VCS commit history started out very detailed and regular but gradually slipped. This can make rollbacks harder, so it is important to keep commits consistent.
- In-code documentation was included, and thorough, but it was not always consistent. Some subroutines were over-commented, some under-commented. Some files followed the correct structure for in-code documentation, others just had English comments.

5.8 LIMITATIONS OF EXISTING SOLUTION

Although the base application is complete (as per requirements) it could be improved by giving it access to a large data set. The current set of data is real-time user tweets from Twitter. However, Twitter limit this to a rate of 3000 tweets per second, which is only a tiny proportion of tweets. This obviously reduces the accuracy of the results.

Secondly, although the application works with a small number of users simultaneously, the server it is hosted on is unlikely to be able to cope with much more than 100 requests per second (about 1000 simultaneous users) for any long period of time. To overcome this, it will need to be migrated onto a more powerful VPS.

Finally, although a huge amount of work went into getting accurate sentiment data, there was a trade-off between accuracy and efficiency. The final sentiment analysis algorithm has an accuracy of about 85%, but it makes up for the lost accuracy in speed, it is 4000 times faster the HP Haven OnDemand (which has an accuracy of 97%). This limitation could be worked on as a further project to explore other techniques of increasing accuracy without affecting time.

5.9 FURTHER WORK AND FUTURE ENHANCEMENTS

The base application is complete, the next steps in improving and developing it further will include:

- Gaining access to a larger set of real-time social media data
- Improving filters to reduce chance of offensive content being displayed
- Migrating the application onto a larger and faster server
- Developing further data visualisations to draw additional trends in the existing data
- More complete documentation, targeted at a wider range of users
- Implementing a more intelligent machine learning algorithm into the backend
- Improving the accuracy of the sentiment analysis algorithm, without impacting on efficiency
- Improving the user interface and site navigation, as well as making it fully mobile compatible
- Marketing the completed solution so it can be found by potential users

5.10 EXAMPLE FINDINGS USING THE FINAL APPLICATION

A set of search terms were made using some random keywords that are relevant to now. It gives an example of how the application can be used to draw trends from search terms, and identify what is causing those trends.

The following trends were found in early April 2016, and are just an example of a small set of use cases that the application can be applied to. Opinions represented here are from Twitter only.

1. London, Manchester and Leeds are the most negative areas of the UK overall
2. Plymouth, Southampton, Bournemouth and Glasgow are the most positive areas of the UK overall
3. People are very positive about 'EasyJet' at Newcastle, Glasgow and Gatwick airports, however they seem to have had a much more negative experience at Bristol, Liverpool and Birmingham airports
4. The keywords that make tweets about 'Cadbury's' positive are 'win', 'heroes' and 'fun', and a trending word in positive tweets is '#CremeEgg'. The keywords in negative tweets include 'sick', 'ill' and 'expensive'
5. The entity extraction page shows that for the search term 'Oxford Brookes' the key places are 'Oxford' and 'London', key professions are 'engineer', 'nurse' and 'architect' and the top TV show/ movie is 'The Night Manager'
6. A top trend in the world right now is '#NationalSiblingsDay' which is 85% positive worldwide, yet only 52% positive in Brazil
7. The highest trend in London right now is 'Cameron' which is negative due to a current news story
8. After a match this morning, 'Liverpool' is 35% more positive than 'Stoke' (Liverpool won)
9. The most positive time of day overall is between 4 and 7pm
10. The most negative time of day for the topic 'tfl' is between 7:30 and 9:30 AM
11. The overall tone analysed for the term 'Trump' is anger (86%) and joy (5%) with 0% of fear
12. The most positive part of the world for the term 'Starbucks' is New York, and the least positive is San Francisco.
13. Currently the most positive British supermarket is 'Tesco' (58%) with the trending words of 'save', 'thanks' and 'tasty'. The least positive supermarket currently is 'Asda' (-12%) with trending words of 'embarrassed', 'limited', 'queue' and 'pay'
14. The parts of the world with the highest volume of tweets in the past 60 minutes is the UK, and the South East cost of America

6 CONCLUSION

6.1 AIM

The main aim of this project was to turn an unworkable mass amount, of personal opinions expressed on Twitter, into useful insights towards the overall feelings and attitudes conveyed for a specific topic or keyword. Further to that it should extract trends between the sentiment data calculated and other factors, such as time or location. All this should be made publically available in real-time, displayed on a clean and simple interface.

This aim was deconstructed further, and a set of requirements in the form of user stories and detailed acceptance criteria. The final solution met 98.5% of these criteria, as well as many extras which were developed during later sprints.

The final application pulls both real-time (general) and semi-historic (for specific topic or search term) data from twitter, calculates the sentiment for each tweet, and displays the results in a series of dynamic and live data visualisations, clearly illustrating trends in the data.

The application solves the problem outlined in the introduction, and meets all its aims.

6.2 FINAL STATS

98.5% of acceptance criteria were met. (1.2% descoped, and 0.3% remaining)

The final user survey showed that using the application was a **99.8%** positive experience

The sentiment analysis module developed has an **85%** accuracy (compared to human calculation)

The sentiment analysis module is **4000** times faster than existing solutions

100% of unit test are passing, and **89%** of the code base is covered by tests

All code has a quality and style score of at least **80%**, and the majority of files are 100%

All dependencies and dev-dependencies are at the latest version (at the time of writing this)

Since the live version of the app was launched (3 weeks ago) it has had **99.5%** up-time

6.3 LINK TO FINAL SOLUTION

The completed fully working published application can be accessed with the following URL:

<http://sentiment-sweep.com/>

The documentation and source code for the application can be found in the Git repo at:

<https://github.com/Lissy93/twitter-sentiment-visualisation>

7 REFERENCES

7.1 RESEARCH REFERENCES

Meesad, P. (2014). Stock trend prediction relying on text mining and sentiment analysis with tweets. Information and Communication Technologies (WICT), 2014 Fourth World Congress on. 11 (3), 257 - 262.

Mahmood, T. Iqbal, T. ; Amin, F. ; Lohanna, W. ; Mustafa, A.. (Dec. 2013). Mining Twitter big data to predict 2013 Pakistan election winner. Multi Topic Conference (INMIC) International. 16 (5), p49 - 54.

Cheng, D. Oculus Inf. Inc., Toronto, ON, Canada Schretlen, P. ; Kronenfeld, N. ; Bozowsky, N. ; Wright, W.. (2013). Tile based visual analytics for Twitter big data exploratory analysis. Big Data, IEEE International Conference on. 8 (3), p2 - 4.

Wenbo Wang Kno.e.sis Center, Wright State Univ., Dayton, OH, USA Lu Chen ; Thirunarayan, K. ; Sheth, A.P.. (2012). Harnessing Twitter "Big Data" for Automatic Emotion Identification. Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Conference on Social Computing (SocialCom). p587 - 592.

Rahnama, A.H.A. Dept. of Math. Inf. Technol., Univ. of Jyväskylä, Jyväskylä, Finland . (2014). Distributed real-time sentiment analysis for big data social streams. Control, Decision and Information Technologies (CoDIT), 2014 International Conference on. p789 - 794.

Wickramaarachchi, C. Kumbhare, A. ; Frincu, M. ; Chelmiss, C. ; Prasanna, V.K.. (4-7 May 2015). Browse Conference Publications > Cluster, Cloud and Grid Compu ... Help Working with Abstracts Back to Results Real-Time Analytics for Fast Evolving Social Graphs. Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on. 1 (1), p829 - 834.

Hamed, A.A. ; Dept. of Comput. Sci., Univ. of Vermont, Burlington, VT, USA ; Xindong Wu. (June 27 2014-July 2 2014). Does Social Media Big Data Make the World Smaller? An Exploratory Analysis of Keyword-Hashtag Networks. Big Data (BigData Congress), 2014 IEEE International Congress on. 1 (1), 454 - 461.

Vu Dung Nguyen ; Big Data Lab., Univ. of St Andrews, St. Andrews, UK ; Varghese, B. ; Barker, A.. (6-9 Oct. 2013). The royal birth of 2013: Analysing and visualising public sentiment in the UK using Twitter. Big Data, 2013 IEEE International Conference on. 1 (1), p46 - 54.

Osman, A.H. ; Fac. of Comput. Sci. & Inf. Syst., Univ. Teknol. Malaysia, Skudai, Malaysia ; Salim, N.. (26-28 Aug. 2013). An improved semantic plagiarism detection scheme based on Chi-squared automatic interaction detection. Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on. 1 (1), p640 - 647.

7.2 METHODOLOGY REFERENCES

Office of Information Services. (2008). Selecting a Development Approach. Centre for Technology in Government. - 3-4.

Linus Torvalds. (2016). Git. Available: <https://git-scm.com/>. Last accessed March 1st.

expressjs. (2014). body-parser. Available: <https://github.com/expressjs/body-parser>. Last accessed August 2015.

jashkenas. (2011). coffee-script. Available: <https://github.com/jashkenas/coffeescript>. Last accessed August 2015.

expressjs/cookie-parser · GitHub. 2015. expressjs/cookie-parser · GitHub. [ONLINE] Available at: <https://github.com/expressjs/cookie-parser>. [Accessed 12 October 2015].

visionmedia/debug · GitHub. 2015. visionmedia/debug · GitHub. [ONLINE] Available at: <https://github.com/visionmedia/debug>. [Accessed 12 October 2015].

strongloop/express · GitHub. 2015. strongloop/express · GitHub. [ONLINE] Available at: <https://github.com/strongloop/express>. [Accessed 12 October 2015].

jadejs/jade · GitHub. 2015. jadejs/jade · GitHub. [ONLINE] Available at: <https://github.com/jadejs/jade>. [Accessed 12 October 2015].

Automattic/mongoose · GitHub. 2015. Automattic/mongoose · GitHub. [ONLINE] Available at: <https://github.com/Automattic/mongoose>. [Accessed 12 October 2015].

expressjs/morgan · GitHub. 2015. expressjs/morgan · GitHub. [ONLINE] Available at: <https://github.com/expressjs/morgan>. [Accessed 12 October 2015].

petehunt/node-jsx · GitHub. 2015. petehunt/node-jsx · GitHub. [ONLINE] Available at: <https://github.com/petehunt/node-jsx>. [Accessed 12 October 2015].

facebook/react · GitHub. 2015. facebook/react · GitHub. [ONLINE] Available at: <https://github.com/facebook/react>. [Accessed 12 October 2015].

Socket.IO. 2015. Socket.IO. [ONLINE] Available at: <http://socket.io/>. [Accessed 12 October 2015].

Node.js. (2016). Node.js Website. Available: <https://nodejs.org/en/>. Last accessed February 2016.

Adrian Mejia. (2014). MEAN Stack - MongoDB ExpressJS AngularJS NodeJS (Part III). Available: <http://adrianmejia.com/blog/2014/10/03/mean-stack-tutorial-mongodb-expressjs-angularjs-nodejs/>. Last accessed 4th January 2016.

Kai Lei ; Yining Ma ; Zhi Tan. (19-21 Dec. 2014). Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js. Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on. 1 (2), 2-5.

Stefan Tilkov ; Steve Vinoski. (Nov.-Dec. 2010). Node.js: Using JavaScript to Build High-Performance Network Programs. IEEE Internet Computing . 14 (6), 80 - 83.

Andres Ojamaa ; Karl D  una. (2014). Assessing the security of Node.js platform. Internet Technology And Secured Transactions, 2012 International Conference for. 2 (1), p348 - 355.

Xiao-Feng Gu ; Int. Centre for Wavelet Anal. & Its Applic., Univ. of Electron. Sci. & Technol. of China, Chengdu, China ; Le Yang ; Shaoquan Wu. (2014). A real-time stream system based on node.js.Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2014 11th International . 11 (1), 479 - 482.

Andrew John Poulter ; Fac. of Eng. & the Environ., Univ. of Southampton, Southampton, UK ; Steven J. Johnston ; Simon J. Cox. (14-16 Dec. 2015). Using the MEAN stack to implement a RESTful service for an Internet of Things application. Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on. x (x), 280 - 285.

TJ Holowaychuk. (2016). *Express*. Available: <http://expressjs.com/>. Last accessed March 2016.

gotwarlost. 2014. Istanbul. [ONLINE] Available at: <https://github.com/gotwarlost/istanbul>. [Accessed 01 March 16].

TJ Holowaychuk. 2011. Mocha JS. [ONLINE] Available at: <https://mochajs.org/>. [Accessed 01 March 16].

The Chai Assertion Library. 2015. Chai JS. [ONLINE] Available at: <http://chaijs.com/>. [Accessed 01 March 16].

Core Less Team. 2015. Less CSS. [ONLINE] Available at: <http://lesscss.org/>. [Accessed 01 March 16].

CoffeeScript. 2015. CoffeeScript. [ONLINE] Available at: <http://coffeescript.org/>. [Accessed 01 March 16].

gulp. 2015. gulp. [ONLINE] Available at: <http://gulpjs.com/>. [Accessed 01 March 16].

NGINX Inc. 2015. NGINX. [ONLINE] Available at: <https://www.nginx.com/>. [Accessed 01 March 16].

The CentOS Project. 2015. CentOS. [ONLINE] Available at: <https://www.centos.org/>. [Accessed 01 March 16].

JetBrains. 2015. WebStorm. [ONLINE] Available at: <https://www.jetbrains.com/webstorm/>. [Accessed 01 March 16].

GitHub. 2015. GitHub. [ONLINE] Available at: <https://github.com/>. [Accessed 01 March 16].

git. 2015. Git. [ONLINE] Available at: <https://git-scm.com/>. [Accessed 01 March 16].

CONTRIBUTORS. 2015. Socket.io. [ONLINE] Available at: <http://socket.io/>. [Accessed 01 March 16].

Node.js Foundation. 2015. Express.js . [ONLINE] Available at: <http://expressjs.com/>. [Accessed 01 March 16].

Node.js Foundation. 2015. Node.js. [ONLINE] Available at: <https://nodejs.org/en/>. [Accessed 01 March 16].

IBM. 2015. IBM Watson. [ONLINE] Available at: <http://www.ibm.com/smarterplanet/us/en/ibmwatson/>. [Accessed 01 March 16].

HP. 2015. Haven OnDemand. [ONLINE] Available at: <https://www.havenondemand.com/>. [Accessed 01 March 16].

Twitter. 2015. Twitter API. [ONLINE] Available at: <https://dev.twitter.com/rest/public>. [Accessed 01 March 16].

Google. 2015. Google Places API. [ONLINE] Available at: <https://developers.google.com/places/>. [Accessed 01 March 16].

7.3 LIST OF TABLES

Table 1	Table showing the likelihood, impact and risk exposure of each identified risk	16
Table 2	Table showing probability against impact of each risk	17
Table 3	Table showing a breakdown of all work to be completed in each timeframe	18
Table 4	Gantt Chart showing time allocated to each development task.....	19
Table 5	Table showing which user stories will be completed in each sprint.....	20
Table 6	Table showing the pass fail criteria for each sprint	25
Table 7	Summary of fetch-tweets module	51
Table 8	Summary of stream-tweets module	51
Table 9	Summary of find-region-from-location module.....	54
Table 10	Summary of place-lookup module	54
Table 11	Summary of tweet-location module	54
Table 12	Summary of sentiment-analysis module.....	55
Table 13	Summary of remove-words module	55
Table 14	User experience survey 1	81
Table 15	User Experience survey 2	82
Table 16	User experience survey 3	83
Table 17	Final evaluation of user stories	89

7.4 LIST OF FIGURES

Fig 1	Dictionary Vs Machine learning SA	10
Fig 2	Scatter chart showing accuracy of various SA methods	13
Fig 3	Radar chart showing conclusion of different SA methods.....	14
Fig 4	Wireframe for home screen.....	29
Fig 5	Wireframe for the heat map screen	30
Fig 6	Wireframe for the region map screen	30
Fig 7	Wireframe for the timeline screen	31
Fig 8	Wireframe for the raw tweets screen.....	31
Fig 9	Wireframe for the globe screen.....	31
Fig 10	Wireframe for the word cloud screen.....	32
Fig 11	Wireframe for the word scatter plot screen	32
Fig 12	Wireframe for the trending screen	33
Fig 13	Wireframe for the comparison screen.....	33
Fig 14	Wireframe for the entity extraction screen	34
Fig 15	Wireframe for the tone identification screen.....	34
Fig 16	Wireframe for the search page	35
Fig 17	Data flow diagram for the backend as a whole	37
Fig 18	Traditional server (compared to Node approach)	38
Fig 19	Node server (compared to traditional approach).....	38
Fig 20	The Node.js processing model	39
Fig 21	NGINX architecture	40
Fig 22	Screenshot of heat map	57
Fig 23	Screen shot of 3D globe	59
Fig 24	Screen shot of regional map	60
Fig 25	Screen shot of word cloud	61
Fig 26	Screen shot of word scatter plot.....	61
Fig 27	Screen shot of time line	65
Fig 28	Screen shot of trending 1	66
Fig 29	Screen shot of trending 2	67
Fig 30	Data flow diagram for trending page.....	68
Fig 31	Screen shot for comparison	70
Fig 32	Diagram of node socket real-time.....	71
Fig 33	Screen shot of the Now page	71
Fig 34	Screen shot of the tone identification page.....	72
Fig 35	Screen shot of the entity extraction page.....	73
Fig 36	Screen shot of the home page	74
Fig 37	Screen shot of second part of home page	75
Fig 38	Screen shot of the search page	76
Fig 39	Logo	80

1 APPENDIX ONE

USER STORIES

This appendix document describes the solutions functional requirements in the form of user stories, along with acceptance criteria and complexity estimates.

Twitter Sentiment Visualisations

The research and development of a sentiment analysis module, and the implementation of it on real-time social media data, to generate a series of live visual representations of sentiment towards a specific topic or by location in order to find trends.

1.1 CONTENTS

1	Appendix One User Stories.....	102
1.1	Contents.....	102
1.2	Overview of User Stories.....	103
1.3	Acceptance Criteria.....	109
1.4	Story Points.....	122
1.5	Summary.....	129

Alicia Sykes 12011471

Oxford Brookes University

1.2 OVERVIEW OF USER STORIES

1.2.1 Main Programme

USER STORIES RELATING TO THE WHOLE PROGRAMME IN GENERAL

TSV-A001 As a user I should be able to view the solution in my browser simply by visiting a URL

TSV-A002 As a user I should have fast loading times or be displayed with a loading graphic

TSV-A003 As a user I should be able to easily switch between different parts of the application

TSV-A004 As a user I should find the user interface clear and simple to look at

1.2.2 Map

USER STORIES RELATING TO THE DISPLAYING AND FUNCTIONALITY OF THE MAP

TSV-B005 As a user I should be able to view the map

TSV-B006 As a user I should be able to pan (move the map with pointing device) and zoom (with both control buttons and hardware such as mouse wheel)

TSV-B007 As a user I should be able to search for a specific location

TSV-B008 As a user I should find the map clear and neat to look at.

1.2.3 Heat Map

USER STORIES RELATING TO THE DISPLAYING OF THE HEAT MAP OVERLAY

TSV-C009 As a user I should be able to see the heat map over the top of the standard map

TSV-C010 As a user I should be able to toggle the display of the heat map to show and hide it

TSV-C011 As a user I should be able to adjust the opacity of the heat map with a simple slider

TSV-C012 As a system I should bind data to the heat map

TSV-C013 As a system I should be able to bind additional data to the map to update it after the initial heat map has already been rendered

TSV-C014 As a user I should find the colors of the heat map clearly represent the data it is displaying

1.2.4 Sentiment Analysis

USER STORIES RELATING TO THE DETECTION OF SENTIMENT IN TWEETS

TSV-D015 As a system I should be able to identify whether a string is positive, neutral or negative overall

TSV-D016 As a system I should be able to identify whether the attitude towards a certain specified topic is positive, neutral or negative

TSV-D017 As a system I should be able to give a confidence estimate showing how likely it is that the sentiment value is accurate

TSV-D018 As a system I should be able to return emotion analysis results for a given string

TSV-D019 As a system the load time for fetching results should be sufficiently fast

TSV-D020 As a system I should give a suitable response and error code if no result is available

1.2.5 Fetching Tweets

USER STORIES RELATING TO THE PULLING OF TWEETS FROM THE TWITTER API

TSV-E021 As a system I should be able to fetch a specified number of Tweets from around a given location from the Twitter API

TSV-E022 As a system I should be able to fetch a specified number of Tweets talking about a specific topic, hashtag or keyword from the Twitter API

TSV-E023 As a system I should have reasonable response times in fetching Tweets from the Twitter API

TSV-E024 As a system I should respond with a suitable error if there is a problem fetching Tweets from the Twitter API

TSV-E025 As a system I should be able to create a series of Tweet objects ready for the next stage

1.2.6 Displaying Results

BRINGING TOGETHER THE MAP, HEAT MAP, SENTIMENT ANALYSIS AND TWEETS

TSV-F026 As a user I should be able to see the heat map bound to data to display results

TSV-F027 As a user I should be able to see search for a keyword and location to see the heat map for just that topic

TSV-F028 As a user I should be able to hover over specific areas of interest and see what is trending or causing that particular heat patch

TSV-F029 As a system I should be displaying data both from the database and live from Tweets

TSV-F030 As a system I should be able to load fast and display suitable loading graphic when loading times are more than 1 second

1.2.7 Caching

Stories relating to the data caching and database

TSV-G031 As a system I should be able to write to the database asynchronously

TSV-G032 As a system I should be able to read from the database asynchronously

TSV-G033 As a system I should be able to query the database for just relevant Tweets

TSV-G034 As a system I should delete the oldest entries when there reaches a certain number of records

TSV-G035 As a system I should be able to check and make safe data before inserting

TSV-G036 As a system I should use both the cached and live data in conjunction or depending on user selection

1.2.8 Home Screen

Stories relating to the initial landing page of the application

TSV-H037 As a user I should find the start page clear and concise

TSV-H038 As a user I can make a search directly from the home page

TSV-H039 As a user I can navigate to any other part of the application directly from the home screen

1.2.9 Regional Map Screen

Stories relating to the regional map front-end

TSV-I040 - As a user I should be able to get an immediate overview of results

TSV-I041 – As a user I should be able search for a specific search term

TSV-I042 – As a user I should be able to see a list of most positive and negative regions

1.2.10 Trending Screen

Stories relating to the trending topics front-end

TSV-J043 - As a user I should be able see what is currently trending worldwide

TSV-J044 – As a user I should be able to enter a custom location to view local trends

TSV-J045 – As a user I can click a trend to find out more

TSV-J046 – As a user I can view trends visually, to get an overview of volume and sentiment

1.2.11 Text Tweets Screen

Stories for the front-end of the text tweets screen, which shows displays raw tweets

TSV-K047 – As a user I should be able to read the plain text tweets relating to a chosen topic/ word

TSV-K048 – As a user I should be able to quickly pick out the key information while reading

TSV-K049 – As a user I should see new tweets come in in real-time

1.2.12 Word Cloud

Stories for the front-end of the word cloud screen

TSV-L050 – As a user I can see a word cloud after entering my search term

TSV-L051 – As a user I can see a text list of the top words used in positive and negative tweets

1.2.13 Word Scatter Plot

Stories for the front-end of the word scatter plot screen

TSV-M052 – As a user I can view the scatter plot for a given search term/ keyword

TSV-M053 – As a user I can hover over a point to view the associated key word

1.2.14 3D Sentiment Globe

Stories for the front-end of the 3D sentiment globe

TSV-N054 – As a user I can interact with the globe

TSV-N055 – As a user I can see sentiment for each location (that has Twitter) on the globe

TSV-N056 – As a user I can view statistics of the data shown on the globe

1.2.15 Timeline

User stories relating to the sentiment over time-of-day visualisation

TSV-O057 – As a user I can see both positive and negative sentiment over the past 24 hours

TSV-O058 – As a user I can interact with the map and the axis

1.2.16 Comparison

User stories relating to the topic comparison page

TSV-P059 – As a user I can enter between one and four search terms

TSV-P060 – As a user I can see the overall sentiment for each of my search terms

TSV-P061 – As a user I can see most commonly used words and their sentiment for each topic

TSV-P062 – As a user I can view links to the other data visualisations for each of the topics

1.2.17 Tone Identification

User stories relating to screen that identifies tone of language of tweets

TSV-Q063 – As a user I can view a summary of the key tones identified for a topic

TSV-Q64 – As a user I can see a more detailed breakdown in the form of a segmented radar chart

1.2.18 Entity Extraction

User stories relating to the screen that displays the entity extraction results

TSV-R065 – As a user I can view the key entities extracted from a set of tweets

TSV-R066 – As a user I can see what volume of each entity and category visually on a Sankey chart

1.2.19 Search Screen

User stories relating the search results screen

TSV-S067 – As a user I can make a search

TSV-S068 – As a user I can get a quick overview of the average sentiment of my topic

TSV-S069 – As a user I can see which keywords are most commonly used in the twitter results

TSV-S070 – As a user I can see a summary of the tones identified

TSV-S071 – As a user I can see a summary of the entities extracted

TSV-S072 – As a user I can see how my topic compares to the rest of Twitter

TSV-S073 – As a user I can read a set of tweets relating to my topic

TSV-S074 – As a user I can view more results for my search term on each data visualisation

1.2.20 Development Environment

User stories relating to how code is managed in the dev environment

TSV-T075 – As a development environment I should compile CoffeeScript into JavaScript

TSV-T076 – As a development environment I should compile LESS/ SASS into CSS

TSV-T077 – As a development environment I should check HTML/ Jade for missing entities

TSV-T078 – As a development environment I should prepare graphics from source to their web form

TSV-T079 – As a development environment I should remove obsolete files

TSV-T080 – As a development environment I should bundle browser scripts

TSV-T081 – As a development environment I should check code (where possible) for errors /bad style

TSV-T082 – As a development environment I should watch for changes in source files and then recompile the appropriate files

TSV-T083 – As a development environment I should keep multiple development browsers across various devices in sync for seamless compatibility testing

TSV-T084 – As a development environment I should run unit tests and output results

1.2.21 Testing

User stories relating to the test environment

TSV-U085 – As a test environment I should run unit tests

TSV-U086 – As a test environment I should have integration tests

TSV-U087 – As a test environment I should be able to stub data as to not make external data requests

TSV-U088 – As a test environment I should use assertions for the unit tests

TSV-U089 – As a test environment I should show coverage test results

TSV-U090 – As a test environment I should check dependencies are up-to-date

TSV-U091 – As a test environment I should check code quality with automated code reviews

TSV-U092 – As a test environment I should do headless browser testing and HTTP service testing

1.2.22 Real-time data streaming

User stories relating to the Twitter streaming module

TSV-V093 – As a developer I should be able to get a continuous stream of chunked tweets

TSV-V094 – As a developer, the streamed tweets should be formatted and safe

1.2.23 Keyword Analysis

User stories relating to the extraction of keywords

TSV-W095 – As a developer I should be able to extract an array of keywords from a string

TSV-W096 – As a developer, I should be able to customise the deletion word set

1.2.24 Geographical Place Lookup

User stories relating to the geographical place modules

TSV-X097 - As a developer I can get a valid latitude and longitude for any fuzzy place name

TSV-X098 – As a developer I can get the country code in multiple forms from latitude and longitude

TSV-X099 – As a developer I can get location of geo-tagged tweets from their ID's

1.2.25 Efficiency

User stories relating to system efficiency

TSV-Z100 – As a system I am super-efficient (and totally awesome)

1.3 ACCEPTANCE CRITERIA

1.3.1 Main Programme

USER STORIES RELATING TO THE WHOLE PROGRAMME IN GENERAL

TSV-A001 As a user I should be able to view the solution in my browser simply by visiting a URL

- AC1 There will be no login, start screen or any other pages to navigate through before being presented with the solution
- AC2 The URL will use the http protocol
- AC3 The web page must be compatible with modern browsers including Chrome, Firefox and Safari
- AC4 The web page must be indexed and crawlable by search engines

TSV-A002 As a user I should have fast loading times or be displayed with a loading graphic

- AC1 The initial page should load in under 1 second
- AC2 A loading animation should be displayed for anything that takes more than 1 second
- AC3 Where possible loading status should be displayed
- AC4 The visualisation should be displayed as soon as there is sufficient data loaded

TSV-A003 As a user I should be able to easily switch between different parts of the application

- AC1 There should be several clear buttons visible at the top of the page
- AC2 The user should only have to click once to change view
- AC3 If data needs to be loaded then a suitable loading graphic should be displayed

TSV-A004 As a user I should find the user interface clear and simple to look at

- AC1 The visualisation should be full screen with the only other component on the screen being the navigation bar at the top
- AC2 There should be no unnecessary information displayed by default

1.3.2 Map

USER STORIES RELATING TO THE DISPLAYING AND FUNCTIONALITY OF THE MAP

TSV-B005 As a user I should be able to view the map

- AC1 The map will be shown by default
- AC2 The map will display consistently in all modern browsers

TSV-B006 As a user I should be able to pan (move the map with pointing device) and zoom (with both control buttons and hardware such as mouse wheel)

- AC1 There will be a maximum and minimum field of view defined for zoom
- AC2 The user will be able to use their default pointing device to navigate the map

TSV-B007 As a user I should be able to search for a specific location

- AC1 The search box will be clear and located at the top of the screen
- AC2 The search box will have a placeholder text clearly indicating what should be entered

- AC3 The search box will have autocomplete using Google Places. When the user starts typing a dropdown of matching places will be displayed similar to that on the Google.com map
- AC4 The user should be able to search by postcode, address, place name, area, district or latitude & longitude

TSV-B008 As a user I should find the map clear and neat to look at.

- AC1 The map should be pale-greyscale colour
- AC2 The map should only have labels for significant place names
- AC3 More places should be displayed as the user zooms in, and the detail decreased when the user zooms out

1.3.3 Heat Map

USER STORIES RELATING TO THE DISPLAYING OF THE HEAT MAP OVERLAY

TSV-C009 As a user I should be able to see the heat map over the top of the standard map

- AC1 The heat map should not affect the functionality of the underlying map
- AC2 The heat map overlay should be semi-transparent to show the map underneath

TSV-C010 As a user I should be able to toggle the display of the heat map to show and hide it

- AC1 The toggle button should display as a simple switch
- AC2 The overlay should disappear and reappear instantly

TSV-C011 As a user I should be able to adjust the opacity of the heat map with a simple slider

- AC1 The opacity should range from 0 to 1
- AC2 There should be no lag

TSV-C012 As a system I should bind data to the heat map

- AC1 The data source the heat map reads from should be what is outputted by the backend system

TSV-C013 As a system I should be able to bind additional data to the map to update it after the initial heat map has already been rendered

- AC1 The system should be constantly looking for changes in the database
- AC2 The system should be constantly waiting for new updates from the Twitter module
- AC3 The updates should be seamless with no visible loading or lag

TSV-C014 As a user I should find the colors of the heat map clearly represent the data it is displaying

- AC1 There should be a legend indicating what color is what

1.3.4 Sentiment Analysis

USER STORIES RELATING TO THE DETECTION OF SENTIMENT IN TWEETS

TSV-D015 As a system I should be able to identify weather a string is positive, neutral or negative overall

- AC1 A request should be sent containing a string
- AC2 The string will be sent to relevant API
- AC3 A response will be returned from the API and processed by the module

TSV-D016 As a system I should be able to identify weather the attitude towards a certain specified topic is positive, neutral or negative

- AC1 An additional parameter containing the keyword needs to be passed in
- AC2 The response must be relevant to that keyword

TSV-D017 As a system I should be able to give a confidence estimate showing how likely it is that the sentiment value is accurate

- AC1 This should be expressed as a decimal between 0 and 1
- AC2 This is generated by the API

TSV-D018 As a system I should be able to return emotion analysis results for a given string

- AC1 The system should return the list of emotions, each with a decimal value between 0 and 1 indicating how much of that emotion was detected
- AC2 If none of an emotion is found it will return 0

TSV-D019 As a system the load time for fetching results should be sufficiently fast

- AC1 Load time must be under 1 second
- AC2 On the frontend suitable load animations should be displayed

TSV-D020 As a system I should give a suitable response and error code if no result is available

- AC1 The right error code should be returned
- AC2 All result fields should be returned just with empty values

1.3.5 Fetching Tweets

USER STORIES RELATING TO THE PULLING OF TWEETS FROM THE TWITTER API

TSV-E021 As a system I should be able to fetch a specified number of Tweets from around a given location from the Twitter API

- AC1 The given location must be converted to a suitable unit
- AC2 Only tweets from around that location should be displayed
- AC3 Results must be in JSON format
- AC4 Calls must be made asynchronously

TSV-E022 As a system I should be able to fetch a specified number of Tweets talking about a specific topic, hashtag or keyword from the Twitter API

- AC1 The keyword should be between 2 and 25 characters long and stripped of all special characters
- AC2 Only Tweets containing the given keyword should be returned

- AC3 Results must be in JSON format
- AC4 Calls must be made asynchronously

TSV-E023 As a system I should have reasonable response times in fetching Tweets from the Twitter API

- AC1 – There should be as few requests as physically possible
- AC2 – Only data that is going to be used should be requested where possible
- AC3 – If a data request takes more than 4 seconds it should time out and a backup data source used or show appropriate error message

TSV-E024 As a system I should respond with a suitable error if there is a problem fetching Tweets from the Twitter API

- AC1 An error code and clear message should be displayed
- AC2 There should be a troubleshooting section with a brief description and possible resolution of each error message and code

TSV-E025 As a system I should be able to create a series of Tweet objects for the next stage

- AC1 Each Tweet object should follow the data structure outlined in the data structures section
- AC2 The sentiment value should be left blank at this stage

1.3.6 Displaying Results

BRINGING TOGETHER THE MAP, HEAT MAP, SENTIMENT ANALYSIS AND TWEETS

TSV-F026 As a user I should be able to see the heat map bound to data to display results

- AC1 The heat map created will use the data fetched from the sentiment analysis module

TSV-F027 As a user I should be able to search for a keyword and location to see the heat map for just that topic

- AC1 The map view must update if a location is searched for
- AC2 If a specific location is searched for fresh tweets will be fetched
- AC3 Twitter results will be added to the map as they come on, rather than waiting for the whole batch to finish first
- AC4 If a keyword is searched, only tweets resulting to that keyword will be displayed

TSV-F028 As a user I should be able to hover over specific areas of interest and see what is trending or causing that particular heat patch

- AC1 Once the users mouse has been still for more than 0.4 seconds a tooltip will display
- AC2 The tooltip can be clicked for further details
- AC3 The expanded tooltip will show a list of trending hashtags around the place of interest
- AC4 If a hashtag is clicked then the map will update to show just the attitudes towards that hashtag

TSV-F029 As a system I should be displaying data both from the database and live Tweets

- AC1 The system must be able to watch for changes on Twitter
- AC2 The system must be able to trigger a call when there is a change
- AC3 The call must fetch the tweet and process it
- AC4 The UI must be updated without a complete re-render

TSV-F030 As a system I should be able to load fast and display suitable loading graphic when loading times are more than 1 second

- AC1 A turning animation will be displayed for any loads more than 1 second
- AC2 Where possible the load status will be displayed
- AC3 The loading will time out if any one task takes more than 4 seconds to complete. A suitable error will be shown.

1.3.7 Caching

Stories relating to the data caching and database

TSV-G031 As a system I should be able to write to the database asynchronously

- AC1 The system will be able to write JSON data following a schema to the database
- AC2 The system will be able to write data while it is doing other tasks
- AC3 The writing to the database will not hold up or block any other threads

TSV-G032 As a system I should be able to read from the database asynchronously

- AC1 The system will be able to read JSON data following a schema from the database
- AC2 The system will be able to read data while it is doing other tasks
- AC3 The reading from the database will not hold up or block any other threads

TSV-G033 As a system I should be able to query the database for just relevant Tweets

- AC1 Parameters including keywords, handle, location, time should all be able to be queried

TSV-G034 As a system I should delete the oldest entries when there reaches a certain number of records

- AC1 research the optimum number of Tweets to store in the database based on efficient rendering of the map and enough detail to make the basic visualisation effective
- AC2 When the number of Tweets reaches this number start to delete older Tweets whenever a new one is inserted

TSV-G035 As a system I should be able to check and make safe data before inserting

- AC1 All unnecessary data, such as URL's, pictures and additional handles will be removed
- AC2 All special characters will be escaped
- AC3 The maximum length per Tweet will be cut to 145 characters

TSV-G036 As a system I should use both the cached and live data in conjunction or depending on user selection

- AC1 If the user has searches for a location or hashtag, while it is loading database data will be displayed, and the map will then update according to new Tweets

1.3.8 Home Screen

Stories relating to the initial landing page of the application

TSV-H037 As a user I should find the start page clear and concise

- AC1 It will not contain more information than necessary
- AC2 It will give a very brief overview about what the application is, and what it can be used for

TSV-H038 As a user I can make a search directly from the home page

- AC1 There will be a search field to enter a keyword or topic
- AC2 The user will be redirected to the search results page to display the findings
- AC3 It should be clear to the user where to search and what to enter

TSV-H039 As a user I can navigate to any other part of the application directly from the home screen

- AC1 There will be a link to each section of the website, including all 10 data visualisations
- AC2 There will also be a link to the `about` page and the source code and documentation

1.3.9 Regional Map Screen

Stories relating to the regional map front-end

TSV-I040 - As a user I should be able to get an immediate overview of results

- AC1 Should show geographical sentiment towards a specified topic
- AC2 Each region should be a single colour, with no gradients
- AC3 A region should be clickable
- AC4 When user hovers or clicks a region, they should be able to view numeric results

TSV-I041 – As a user I should be able search for a specific search term

- AC1 There must be an input box
- AC2 Results must then be rendered according to the value of the search field

TSV-I042 – As a user I should be able to see a list of most positive and negative regions

- AC1 After the user has entered their topic and the map has been rendered, a bullet point list of the most positive regions, and most negative regions towards the specified topic.
- AC2 Items in the list must display a sentiment value in percentage form
- AC3 Items in the list should be clickable

1.3.10 Trending Screen

Stories relating to the trending topics front-end

TSV-J043 - As a user I should be able see what is currently trending worldwide

- AC1 A list of the top ten trends worldwide will be shown
- AC2 The sentiment of each will be indicated by colour
- AC3 The list will be sorted by volume of tweets, with the most popular trends at the top

TSV-J044 – As a user I should be able to enter a custom location to view local trends

- AC1 - An input field will be provided for the user to enter a location
- AC2 The user should be able to enter any place e.g. 'OX3 0BP', '14 Greys Road', 'California', 'Europe'. The Google Places API will then be used to find the latitude and longitude of the given string.
- AC3 The trends displayed should be specific to the users location

TSV-J045 – As a user I can click a trend to find out more

- AC1 The user should be able to click any trend and be redirected to the search page
- AC2 It should be clear to the user the trends are clickable
- AC3 Trends in the bubble chart (see below) will also be clickable

TSV-J046 – As a user I can view trends visually, to get an overview of volume and sentiment

- AC1 There will be a bubble diagram below the list of trends
- AC2 The diagram will show one bubble for each trend
- AC3 The size of the bubble will represent the volume of tweets for that trend
- AC4 The colour of each bubble will indicate sentiment
- AC5 The bubble will be clickable to gain more information about a given trend

1.3.11 Text Tweets Screen

Stories for the front-end of the text tweets screen, which shows displays raw tweets

TSV-K047 – As a user I should be able to read the plain text tweets relating to a chosen topic/ word

- AC1 There must be a search field for the user to enter their topic or keyword into
- AC2 Once they press enter fresh data should be fetched and rendered
- AC3 Tweets should be laid out in a clear and concise manor

TSV-K048 – As a user I should be able to quickly pick out the key information while reading

- AC1 Positive and negative tweets should be in separate columns
- AC2 Each tweet should have their keywords in bold, to make scan reading easier
- AC3 The sentiment, date/time and location of each tweet should be displayed

TSV-K049 – As a user I should see new tweets come in in real-time

- AC1 – The user shouldn't have to refresh page to see new data

1.3.12 Word Cloud

Stories for the front-end of the word cloud screen

TSV-L050 – As a user I can see a word cloud after entering my search term

- AC1 Words should vary in size depending on number of times used
- AC2 Sentiment should be represented by the shade of colour, red = negative, green = positive, grey = neutral and all colours in between show sentiment in between

TSV-L051 – As a user I can see a text list of the top words used in positive and negative tweets

- AC1 There should be two lists, one for top words used in positive tweets, the other negative
- AC2 Each word should have an average sentiment value for associated tweets, in percentage
- AC3 Each word should be followed by a number, of how many times it occurred in that set
- AC4 The words should be clickable to view more

1.3.13 Word Scatter Plot

Stories for the front-end of the word scatter plot screen

TSV-M052 – As a user I can view the scatter plot for a given search term/ keyword

- AC1 Sentiment should be displayed along the y-axis
- AC2 Frequency along the x-axis
- AC3 The colour of the points should also be relative to the sentiment

TSV-M053 – As a user I can hover over a point to view the associated key word

- AC1 The word should be displayed as a tooltip
- AC2 The current point should slightly change colour/ size to make clear which one it is
- AC3 The user should also be able to click a point to search for that word

1.3.14 3D Sentiment Globe

Stories for the front-end of the 3D sentiment globe

TSV-N054 – As a user I can interact with the globe

- AC1 The user should be able to rotate the globe
- AC2 The user should be able to zoom in and out of the globe

TSV-N055 – As a user I can see sentiment for each location (that has Twitter) on the globe

- AC1 There should be vertical bars for each settlement with Twitter
- AC2 The colour of the bar will represent sentiment
- AC3 The height of the bar will represent scale of sentiment and volume of tweets

TSV-N056 – As a user I can view statistics of the data shown on the globe

- AC1 Including number of tweets displayed and average sentiment
- AC2 There will also be links to view the same data in other forms

1.3.15 Timeline

User stories relating to the sentiment over time-of-day visualisation

TSV-O057 – As a user I can see both positive and negative sentiment over the past 24 hours

- AC1 The positive sentiment will be represented by a green line and shaded area
- AC2 The negative sentiment will be represented by a red line and shaded area

TSV-O058 – As a user I can interact with the map and the axis

- AC1 The user should be able to hover over the line to see the sentiment at a given time
- AC2 The user should be able to hover over the line to see time for a given sentiment value

1.3.16 Comparison

User stories relating to the topic comparison page

TSV-P059 – As a user I can enter between one and four search terms

- AC1 The screen will be divided into the number of search terms entered
- AC2 The user should not be able to enter over four topics, or zero topics

TSV-P060 – As a user I can see the overall sentiment for each of my search terms

- AC1 this should be displayed in the form of a donut chart for each topic
- AC2 the chart should be labelled, and the user can hover over it to get more details

TSV-P061 – As a user I can see most commonly used words and their sentiment for each topic

- AC1 the top ten words will be displayed
- AC2 the words will be coloured according to sentiment
- AC3 the words will be clickable

TSV-P062 – As a user I can view links to the other data visualisations for each of the topics

1.3.17 Tone Identification

User stories relating to screen that identifies tone of language of tweets

TSV-Q063 – As a user I can view a summary of the key tones identified for a topic

- AC1 these should be displayed in the form of no more than 12 mini bars on a chart
- AC2 the user should be able to hover over a bar to get a more accurate percentage value

TSV-Q64 – As a user I can see a more detailed breakdown in the form of a segmented radar chart

1.3.18 Entity Extraction

User stories relating to the screen that displays the entity extraction results

TSV-R065 – As a user I can view the key entities extracted from a set of tweets

- AC1 Entities should be grouped into categories
- AC2 The first few matches for each entity should be displayed
- AC3 Where possible an image should be shown for each entity, pulled from Wikipedia

TSV-R066 – As a user I can see what volume of each entity and category visually on a Sankey chart

1.3.19 Search Screen

User stories relating the search results screen

TSV-S067 – As a user I can make a search

- AC1 Either from the homepage, initial search page or search results page

TSV-S068 – As a user I can get a quick overview of the average sentiment of my topic

- AC1 A gauge should show average sentiment
- AC2 A hexagon mesh should give a quick overview of sentiment of individual tweets

TSV-S069 – As a user I can see which keywords are most commonly used in the twitter results

- AC1 Should be listed in order of volume of use
- AC2 Should be highlighted according to sentiment
- AC3 Should be clickable

TSV-S070 – As a user I can see a summary of the tones identified

- AC1 Each tone identified and the percentage certainty should be displayed
- AC2 Displayed in the form of bars
- AC3 User should be able to hover to see exact percent
- AC4 There should be a see more button, directing the user to the tone identification page

TSV-S071 – As a user I can see a summary of the entities extracted

- AC1 the top six categories for that topic should be displayed
- AC2 the entities in each six categories should be shown
- AC3 if the user hovers over an entity they should be able to see number of occurrences
- AC4 the entities should be clickable
- AC5 if the system has found an associated Wikipedia image for the entity, it should be displayed
- AC6 there should be a read more button to find and display more entities on the entity screen

TSV-S072 – As a user I can see how my topic compares to the rest of Twitter

- AC1 Should be displayed as a donut chart
- AC2 There should be three input fields to enter more topics to compare with search term
- AC3 Should provide a link to the comparison screen

TSV-S073 – As a user I can read a set of tweets relating to my topic

- AC1 The top five positive tweets should be shown in one column
- AC2 The top five negative tweets should be shown in another column
- AC3 There should be a load more button, to show more tweets from topic
- AC4 The sentiment for each tweet should be shown
- AC5 The location (if applicable) for each tweet should be shown as small grey text
- AC6 The time tweets should be displayed in a readable form in small grey text
- AC7 The keywords of each tweet should be highlighted in bold, and clickable

TSV-S074 – As a user I can view more results for my search term on each data visualisation

- AC1 A thumbnail with image for each applicable data visualisation should be shown
- AC2 Each page will show relevant results to that search term

1.3.20 Development Environment

User stories relating to how code is managed in the dev environment

TSV-T075 – As a development environment I should compile CoffeeScript into JavaScript

- AC1 Should only accept valid CoffeeScript
- AC2 If there's an error in the source code, a suitable message should be outputted
- AC3 Suggestions on how to improve code quality, efficiency and readability should be printed to the console
- AC4 Should only compile code that has changed

TSV-T076 – As a development environment I should compile LESS/ SASS into CSS

- AC1 Should only accept valid LESS/SASS
- AC2 Should give a suitable error message if there's an error in LESS/SASS
- AC3 Should output code quality improvements to the console
- AC4 Should only compile code that has changed since last compile

TSV-T077 – As a development environment I should check HTML/ Jade for missing entities

TSV-T078 – As a development environment I should prepare graphics from source to their web form

- AC1 Graphics should be an optimum size
- AC2 Graphics should be in a suitable format

TSV-T079 – As a development environment I should remove obsolete files

- AC1 Any file which are not referenced anywhere, and hence not being used

TSV-T080 – As a development environment I should bundle browser scripts

- AC1 All read in files (like text, csv, json...) should be bundled
- AC2 All code should be lint-free
- AC3 Output code should be minified
- AC4 Bundles should be size checked

TSV-T081 – As a development environment I should check code (where possible) for errors /bad style

- AC1 Use appropriate plugins to check for each factor
- AC2 Check good code style
- AC3 Check line length is not over 80 chars
- AC4 Print warnings and code quality tips to console
- AC5 Find unused variables and imports
- AC6 Reject errored code

TSV-T082 – As a development environment I should watch for changes in source files and then recompile the appropriate files

TSV-T083 – As a development environment I should keep multiple development browsers across various devices in sync for seamless compatibility testing

- AC1 Both local browsers and external devices such as mobiles and tablets
- AC2 When the developer scrolls down on one device, all should scroll accordingly
- AC3 When the developer clicks a link/ or interacts with one device, should happen on all

TSV-T084 – As a development environment I should run unit tests and output results

1.3.21 Testing

User stories relating to the test environment

TSV-U085 – As a test environment I should run unit tests

- AC1 This should include automated test running
- AC2 Tests should be able to be run locally or remotely
- AC3 Tests should be able to be run with the npm test command

TSV-U086 – As a test environment I should have integration tests

- AC1 These should be fully automated
- AC2 Should be run on every commit
- AC3 Should be run remotely
- AC4 Should run on many different Node.js versions

TSV-U087 – As a test environment I should be able to stub data as to not make external data requests

- AC1 Using Sinon.js
- AC2 There should be no external requests what so ever

TSV-U088 – As a test environment I should use assertions for the unit tests

- AC1 Clean and neat assertions should be written
- AC2 Datatype, size, value.... Should be asserted

TSV-U089 – As a test environment I should show coverage test results

- AC1 An overall coverage percentage should be outputted after tests have run
- AC2 There should also be a much more detailed coverage breakdown
- AC3 A report should be generated and saved to the reports directory
- AC4 Istanbul will be used

TSV-U090 – As a test environment I should check dependencies are up-to-date

- AC1 On every commit
- AC2 Both production dependencies and dev dependencies
- AC3 A badge should be displayed on the readme.md

TSV-U091 – As a test environment I should check code quality with automated code reviews

- AC1 On every commit
- AC2 Should assign a grade
- AC3 Should display a badge on the readme.md

TSV-U092 – As a test environment I should do headless browser testing and HTTP service testing

1.3.22 Real-time data streaming

User stories relating to the Twitter streaming module

TSV-V093 – As a developer I should be able to get a continuous stream of chunked tweets

- AC1 Sanitised data
- AC2 Non-stop

TSV-V094 – As a developer, the streamed tweets should be formatted and safe

1.3.23 Keyword Analysis

User stories relating to the extraction of keywords

TSV-W095 – As a developer I should be able to extract an array of keywords from a string

- AC1 Should return an array
- AC2 Array should include all words in original string EXCEPT words in the remove list
- AC3 By default the remove list will include every generic word

TSV-W096 – As a developer, I should be able to customise the deletion word set

- AC1 By passing in a custom array of words as the second parameter

1.3.24 Geographical Place Lookup

User stories relating to the geographical place modules

TSV-X097- As a developer I can get a valid latitude and longitude for any fuzzy place name

- AC1 Should return the most likely choice for every string passed in

TSV-X098 – As a developer I can get the country code in multiple forms from latitude and longitude

- AC1 Should return ISO2, ISO3 and country code for every country

TSV-X099 – As a developer I can get location of geo-tagged tweets from their ID's

- AC1 Should take the ID as a parameter, and return a valid place object

1.3.25 Efficiency

User stories relating to system efficiency

TSV-Z100 – As a system I am super-efficient (and totally awesome)

1.4 STORY POINTS

Story Number	Story	Story Point Estimate	Reasoning
TSV-A001	As a user I should be able to view the solution in my browser simply by visiting a URL	1	A hosting solution and URL will be set up in sprint zero. After which files will just need to be updated via ftp.
TSV-A002	As a user I should have fast loading times or be displayed with a loading graphic	3	For each load there needs to be a label with the type, if it less than 1 second there is no need for a loading graphic, anything longer there will need to be a suitable screen developed.
TSV-A003	As a user I should be able to easily switch between different parts of the application	3	The application will be set up as a single page application, but only load data when requested.
TSV-A004	As a user I should find the user interface clear and simple to look at	2	The material design standard will be followed throughout.
TSV-B005	As a user I should be able to view the map	2	The application should allow for the map to meet all the points outlined in the acceptance criteria
TSV-B006	As a user I should be able to pan (move the map with pointing device) and zoom (with both control buttons and hardware such as mouse wheel)	1	This functionality should be built into Google Maps, so will just need to be configured.
TSV-B007	As a user I should be able to search for a specific location	5	The will require integration with the Places API, and an AJAX search as well as the creation and implementation of a suitable autocomplete. A further challenge would be to get this running so fast it appears instant.
TSV-B008	As a user I should find the map clear and neat to look at	2	A custom theme will be written and applied to the map.
TSV-C009	As a user I should be able to see the heat map over the top of the standard map	13	The heat map will need to be created and then configured and displayed.
TSV-C010	As a user I should be able to toggle the display of the heat map to show and hide it	3	A suitable switch will need to be developed and implemented and then bound to the map.

TSV-C011	As a user I should be able to adjust the opacity of the heat map with a simple slider	3	Again a suitable slider control will need to be implemented and should affect the configuration of the heat map
TSV-C012	As a system I should bind data to the heat map	5	The data will need to be in the right structure, and the heat map will need to be expecting that structure.
TSV-C013	As a system I should be able to bind additional data to the map to update it after the initial heat map has already been rendered	5	A potential challenge here will be updating the UI after
TSV-C014	As a user I should find the colors of the heat map clearly represent the data it is displaying	2	Suitable colors will be applied following the acceptance criteria. One potential challenge is defining the colors between, as they may look less clear
TSV-D015	As a system I should be able to identify whether a string is positive, neutral or negative overall	5	An AI API will be used so it will need to be configured, tested and integrated.
TSV-D016	As a system I should be able to identify whether the attitude towards a certain specified topic is positive, neutral or negative	5	This will most likely be a different API, and will also need to be configured, tested and integrated in a similar way.
TSV-D017	As a system I should be able to give a confidence estimate showing how likely it is that the sentiment value is accurate	2	Once the API is configured this should be reasonably straight forward to calculate.
TSV-D018	As a system I should be able to return emotion analysis results for a given string	5	This will involve another call to a API with certain parameters in order to determine which emotions are conveyed
TSV-D019	As a system the load time for fetching results should be sufficiently fast	3	This will require efficient algorithms, async tasks and a minimum amount of network calls.
TSV-D020	As a system I should give a suitable response and error code if no result is available	2	This will involve checking numerous conditions, so is not a simple if statement, that is why the point estimate is 2 rather than 1.
TSV-E021	As a system I should be able to fetch a specified number of Tweets from around a given location from the Twitter API	5	Work will also need to be done to process the location data into the correct format.

TSV-E022	As a system I should be able to fetch a specified number of Tweets talking about a specific topic, hashtag or keyword from the Twitter API	3	This will be done in a similar way to fetching the location Tweets, but with keyword rather than location.
TSV-E023	As a system I should have reasonable response times in fetching Tweets from the Twitter API	2	If all the algorithms are efficient, and minimum number of network calls then this will be quite strait forward or determined by the Twitter API.
TSV-E024	As a system I should respond with a suitable error if there is a problem fetching Tweets from the Twitter API	2	There are several different types of error, all conditions will need to be tested and appropriate display to the user.
TSV-E025	As a system I should be able to create a series of Tweet objects ready for the next stage	1	This should be a fairly straight-forward task.
TSV-F026	As a user I should be able to see the heat map bound to data to display results	3	Providing the heat map was correctly set up, and the data is in the right structure there shouldn't be anything too complex here, however data will be coming in after the initial UI has rendered so this needs to be considered.
TSV-F027	As a user I should be able to see search for a keyword and location to see the heat map for just that topic	13	There are several challenges this story may give. Firstly a new bunch of network requests will need to be sent off and be processed, and then the rendered map will need to be re-rendered without page load. If the map is for a specific location zoom will need to be set as well.
TSV-F028	As a user I should be able to hover over specific areas of interest and see what is trending or causing that particular heat patch	13	A hover listener will need to be configured. Also suitable UI components developed and implemented. Most importantly an algorithm determining what is actually causing those heat patches needs to be developed.
TSV-F029	As a system I should be displaying data both from the database and live from Tweets	5	This sounds simple enough, although it does mean the UI will need to update when more data is available on the

			network without a page refresh.
TSV-F030	As a system I should be able to load fast and display suitable loading graphic when loading times are more than 1 second	2	Algorithms must be efficient, with minimum network calls, and fully asynchronous where possible. The loading graphic can just be a simple gif or a css animation.
TSV-G031	As a system I should be able to write to the database asynchronously	3	The best solution may be to use a pre-written package such as Q as it facilitates the 4 main types of async call required for this story.
TSV-G032	As a system I should be able to read from the database asynchronously	2	This should be slightly less complex than the previous story as it is just reading and the data can be pulled down in any order.
TSV-G033	As a system I should be able to query the database for just relevant Tweets	2	In MongoDB this is very easy, just pass in the query in the form of JSON parameters. It does however mean that all data must be in the correct structure.
TSV-G034	As a system I should delete the oldest entries when there reaches a certain number of records	2	Each record will also have a timestamp, and it is simple to count records, so will not be complex to sort by timestamp then remove the appropriate number.
TSV-G035	As a system I should be able to check and make safe data before inserting	1	A function will be written which does everything outlined in the acceptance criteria, it will then just be called for every record that is going to be inserted.
TSV-G036	As a system I should use both the cached and live data in conjunction or depending on user selection	2	All the challenges faced in this story will have been covered on previous stories.
TSV-H037	As a user I should find the start page clear and concise	2	This will need to be worked on across several sprints, but shouldn't pose as too much of a problem technically
TSV-H038	As a user I can make a search directly from the home page	1	Just a simple form
TSV-H039	As a user I can navigate to any other part of the application directly from the home screen	1	These will just be links. However they will need to be clear and contain a graphic. They should also be generated dynamically for code reuse.

TSV-I040	As a user I should be able to get an immediate overview of results of sentiment by location	3	A D3 regional map will need to be developed. This shouldn't be too challenging given the available resources
TSV-I041	As a user I should be able search for a specific search term, on the regional map screen	1	Simple form redirecting to the sub-route which fetches tweets from a different source, but all frontend code will remain the same
TSV-I042	As a user I should be able to see a list of most positive and negative regions	2	The algorithm which calculates this will need to be very efficient, and results should be displayed clearly
TSV-J043	As a user I should be able see what is currently trending worldwide	8	Several steps, to this, must be async
TSV-J044	As a user I should be able to enter a custom location to view local trends	5	Again, many steps, including a place-lookup. Must be asynchronous
TSV-J045	As a user I can click a trend to find out more	1	Reasonable straightforward to make an SVG component clickable
TSV-J046	As a user I can view trends visually, to get an overview of volume and sentiment	3	D3 bubble chart will need to be developed from scratch
TSV-K047	As a user I should be able to read the plain text tweets relating to a chosen topic/ word	3	Nothing particularly complex here, except managing high volumes of textual data
TSV-K048	As a user I should be able to quickly pick out the key information while reading	2	A script will be written using the remove-words module to highlight keywords in bold
TSV-K049	As a user I should see new tweets come in in real-time	3	Real-time functionality for high volumes of data will need to be efficient
TSV-L050	As a user I can see a word cloud after entering my search term	5	Rendering of word cloud positioning may pose as a challenge
TSV-L051	As a user I can see a text list of the top words used in positive and negative tweets	5	Algorithm will need to sort words from a very large set of tweets
TSV-M052	As a user I can view the scatter plot for a given search term/ keyword	3	Same backend as word cloud
TSV-M053	As a user I can hover over a point to view the associated key word	1	Simple D3 function
TSV-N054	As a user I can interact with the globe	2	Following the globe guide, this shouldn't be too complex

TSV-N055	As a user I can see sentiment for each location (that has Twitter) on the globe	5	Will need to render bars onto globe after initial load
TSV-N056	As a user I can view statistics of the data shown on the globe	2	Should be similar to that of the map
TSV-O057	As a user I can see both positive and negative sentiment over the past 24 hours on the timeline	3	Once timeline is rendered this should be simple
TSV-N058	As a user I can interact with the timeline and the axis	2	Since only simple interaction is required it should be fine
TSV-P059	As a user I can enter between one and four search terms to compare	1	Simple form, with dynamically adding fields
TSV-P060	As a user I can see the overall sentiment for each of my search terms	2	Straightforward to calculate and display. Will need to be done for each topic
TSV-P061	As a user I can see most commonly used words and their sentiment for each topic	3	Similar to that of the word cloud, but for each topic, and in a linear way
TSV-P062	As a user I can view links to the other data visualisations for each of the topics	2	Similar to that of the homepage
TSV-Q063	As a user I can view a summary of the key tones identified for a topic	8	High level of complexity compared with sentiment results
TSV-Q064	As a user I can see a more detailed breakdown in the form of a segmented radar chart	5	Radar chart will need to be interactive, and nested circles may get complex
TSV-R065	As a user I can view the key entities extracted from a set of tweets	13	Again much more complex than sentiment results. Also additional resources (like Wikipedia needs to be requested)
TSV-R066	As a user I can see what volume of each entity and category visually on a Sankey chart	5	Sankey chart will need to be interactive and correctly scaled
TSV-S067	As a user I can make a search	2	Simple but crucial so various potential complexities will be considered
TSV-S068	As a user I can get a quick overview of the average sentiment of my topic	3	Simple summary
TSV-S069	As a user I can see which keywords are most commonly used in the twitter results	3	Needs to be calculated efficiently and displayed clearly
TSV-S070	As a user I can see a summary of the tones identified	2	Just summary from tone route

TSV-S071	As a user I can see a summary of the entities extracted	2	Just summary from the entity route
TSV-S072	As a user I can see how my topic compares to the rest of Twitter	8	Will need to quickly calculate sample for the rest of Twitter
TSV-S073	As a user I can read a set of tweets relating to my topic	2	Just a sub extract from the raw tweet page
TSV-S074	As a user I can view more results for my search term on each data visualisation	1	Just links, with custom search term
TSV-T075	As a development environment I should compile CoffeeScript into JavaScript	2	Gulp task
TSV-T076	As a development environment I should compile LESS/ SASS into CSS	2	Gulp task
TSV-T077	As a development environment I should check HTML/ Jade for missing entities	2	Gulp task
TSV-T078	As a development environment I should prepare graphics from source to their web form	2	Gulp task
TSV-T079	As a development environment I should remove obsolete files	2	Gulp task
TSV-T080	As a development environment I should bundle browser scripts	3	Gulp task using Browserify
TSV-T081	As a development environment I should check code (where possible) for errors /bad style	2	Gulp task
TSV-T082	As a development environment I should watch for changes in source files and then recompile the appropriate files	2	Gulp watch task
TSV-T083	As a development environment I should keep multiple development browsers across various devices in sync for seamless compatibility testing	2	Gulp task with Browser-Sync
TSV-T084	As a development environment I should run unit tests and output results	3	Should also include all other tests
TSV-U085	As a test environment I should run unit tests	2	Can be done through the Gulp test task above
TSV-U086	As a test environment I should have integration tests	2	Once configured, very simple and fully automated

TSV-U087	As a test environment I should be able to stub data as to not make external data requests	3	Using Sinon.js which will require upskilling
TSV-U088	As a test environment I should use assertions for the unit tests	2	With Chai. Goes without saying that assertions are required
TSV-U089	As a test environment I should show coverage test results	2	And needs to generate HTML report. Will use Istanbul
TSV-U090	As a test environment I should check dependencies are up-to-date	1	Very simple with once David-DM is configured
TSV-U091	As a test environment I should check code quality with automated code reviews	1	Again, easy with Codeacy
TSV-U092	As a test environment I should do headless browser testing and HTTP service testing	3	SuperTest is less straightforward
TSV-V093	As a developer I should be able to get a continuous stream of chunked tweets	21	A large backend task
TSV-V094	As a developer, the streamed tweets should be formatted and safe	5	Needs to be efficient as large volume of data
TSV-W095	As a developer I should be able to extract an array of keywords from a string	13	Efficiency again is key
TSV-W096	As a developer, I should be able to customise the deletion word set	3	Reasonably simple, but will need verifying
TSV-X097	As a developer I can get a valid latitude and longitude for any fuzzy place name	13	Will use Google API's and own scripts
TSV-X098	As a developer I can get the country code in multiple forms from latitude and longitude	13	Trigonometry
TSV-X099	As a developer I can get location of geo-tagged tweets from their ID's	8	Uses Twitter API
TSV-Z100	As a system I am super-efficient (and totally awesome)		

1.5 SUMMARY

The user stories laid out in this document specify the minimum requirements for the viable project. Many additional requirements were also covered, but weren't specified in this document.

There was a total of 372 story points, these are a measure of complexity rather than time, but usually each story point took about 3 hours.

2 APPENDIX TWO

STYLE GUIDES

This appendix document briefly outlines the good programming practices that should be followed throughout the development of the project. The purpose of it is to ensure consistency and increase clarity of the code.

Twitter Sentiment Visualisations

The research and development of a sentiment analysis module, and the implementation of it on real-time social media data, to generate a series of live visual representations of sentiment towards a specific topic or by location in order to find trends.

Alicia Sykes 12011471

Oxford Brookes University

2.1 COFFEESCRIPT

Tabs or Spaces

Use spaces only, with 2 spaces per indentation level. Never mix tabs and spaces.

Maximum Line Length

Limit all lines to a maximum of 79 characters.

Blank Lines

Separate top-level function and class definitions with a single blank line.

Separate method definitions inside of a class with a single blank line.

Use a single blank line within the bodies of methods or functions in cases where this improves readability (e.g., for the purpose of delineating logical sections).

Trailing Whitespace

Do not include trailing whitespace on any lines.

Optional Commas

Avoid the use of commas before newlines when properties or elements of an Object or Array are listed on separate lines.

Whitespace in Expressions and Statements

Avoid extraneous whitespace in the following situations: immediately inside parentheses, brackets or braces or immediately before a comma

Naming Conventions

Use `camelCase` (with a leading lowercase character) to name all variables, methods, and object properties. Use `CamelCase` (with a leading uppercase character) to name all classes.

For constants, use all uppercase with underscores: `(CONSTANT_LIKE_THIS)`

Methods and variables that are intended to be "private" should begin with a leading underscore: `(_privateMethod: ->)`

Extending Native Objects

Do not modify native objects.

Exceptions

Do not suppress exceptions.

Miscellaneous

`and` is preferred over `&&`.

`or` is preferred over `||`.

`is` is preferred over `==`.

`isnt` is preferred over `!=`.

`not` is preferred over `!`.

Prefer shorthand notation (`::`) for accessing an object's prototype:

Prefer `@property` over `this.property`.

However, avoid the use of standalone `@`:

Avoid `return` where not required, unless the explicit `return` increases clarity.

Use splats (`...`) when working with functions that accept variable numbers of arguments:

3 APPENDIX THREE

VCS HISTORY

A requirement for the projects development was that a version control system (VCS) would be implemented to track changes. Git was used, and this document contains a summary of the Git commit history as proof of the successful implementation of a VCS.

Twitter Sentiment Visualisations

The research and development of a sentiment analysis module, and the implementation of it on real-time social media data, to generate a series of live visual representations of sentiment towards a specific topic or by location in order to find trends.

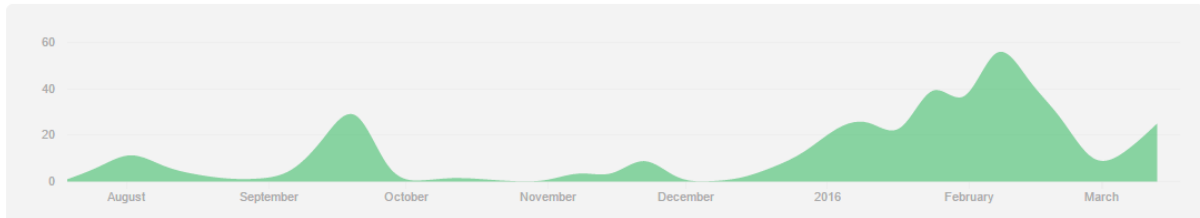
Alicia Sykes 12011471

Oxford Brookes University

3.1 COMMIT GRAPHS

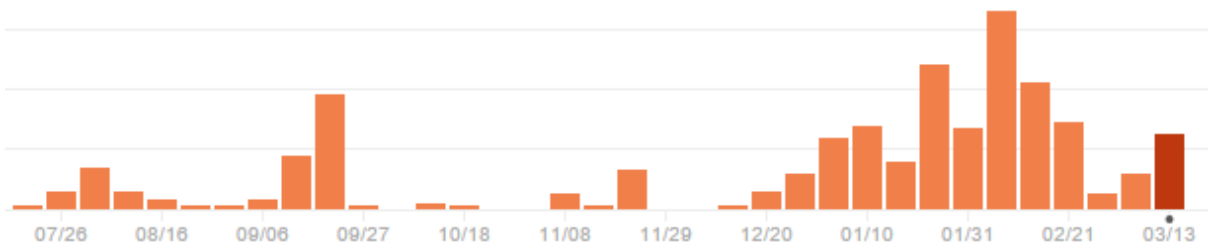
3.1.1 Contributions

The following chart shows the volume of contributions to the repo (y-axis) against time (x-axis) between August 2015 and March 2016.



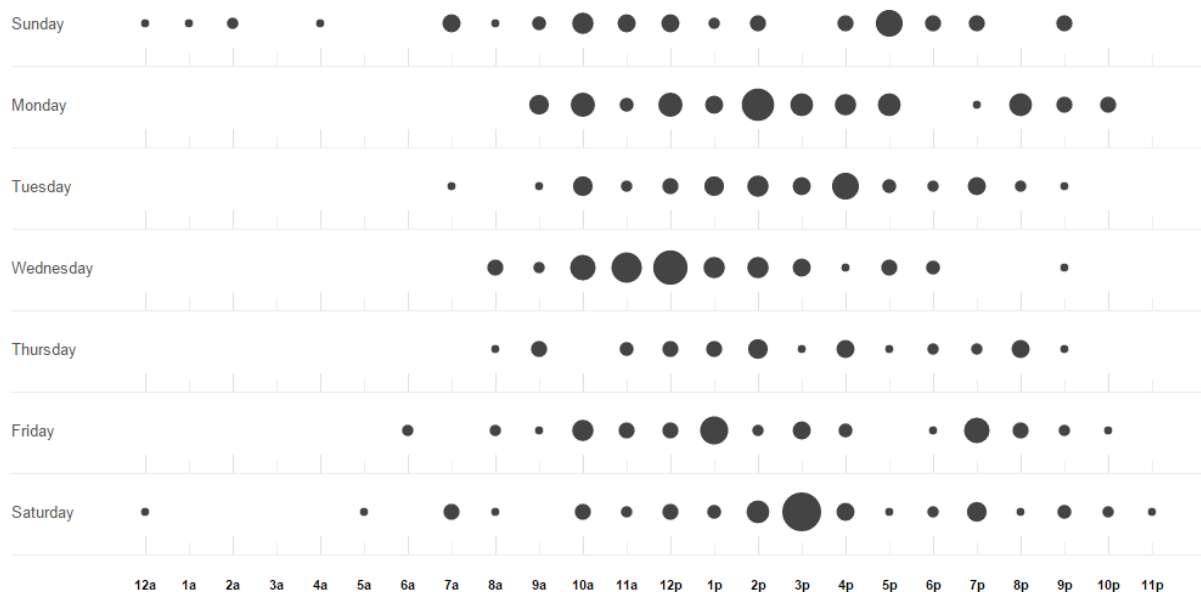
3.1.2 Commits

The commit column chart illustrates the number of commits (y-axis) against time (x-axis).



3.1.3 Punch Card

The punch card chart below shows a summary of the times of day that the most commits were made (if over 15 commits per hour).



3.2 LOGS

3.2.1 How to print

The following command was put together to print out all git commits for the current repository.

To use, you must first have git installed globally on your system, then navigate into the project root, where the .git file is, then simply paste the below command, and the results should be generated. Output will be neatly formatted, and colour coded to make reading easy. Output will be ordered by newest commit first.

```
git log --pretty=format:%Cgreen"%h
%Creset%Cblue[%aD%x08%x08%x08%x08%x08%x08%x08%x08] %Creset %s
%Cred(author: %cn)" -all
```

Example output (from the *find-region-from-position* module)

```
C:\Users\Alicia\Dropbox\Coding\Nodejs\find-region-from-position (master)
λ git log --pretty=format:%Cgreen"%h %Creset%Cblue[%aD%x08%x08%x08%x08%x08%x08%x08%x08] %Creset %s %Cred(author: %cn)" --all
8a981c1 [Wed, 3 Feb 2016 11:07] Wrote some documentation (author: Alicia Sykes)
36b0f54 [Wed, 3 Feb 2016 10:01] Finds and returns closest region, done and working (author: Alicia Sykes)
df79c8b [Wed, 3 Feb 2016 10:00] Wrote some unit tests (author: Alicia Sykes)
f9d604d [Tue, 2 Feb 2016 13:24] Finds a list of closest single directions lat or longs (author: Alicia Sykes)
81633d3 [Tue, 2 Feb 2016 12:46] Wrote convertCsvToJson function (author: Alicia Sykes)
fcaea1 [Tue, 2 Feb 2016 12:46] Minify JS (author: Alicia Sykes)
0701398 [Mon, 1 Feb 2016 20:34] Set up blank module (author: Alicia Sykes)
7a8af49 [Mon, 1 Feb 2016 20:33] Added initial regions lat lng dataset (author: Alicia Sykes)
7e06f93 [Mon, 1 Feb 2016 20:33] Initial project setup NPM init (author: Alicia Sykes)
bb19c08 [Mon, 1 Feb 2016 20:32] Configured Gulp build environment (author: Alicia Sykes)
f49d0c9 [Mon, 1 Feb 2016 20:32] Added MIT License (author: Alicia Sykes)
41f3803 [Mon, 1 Feb 2016 20:31] Configured test environment, with Mocha, Travis, Chai and Istanbul (author: Alicia Sykes)
2aa273b [Mon, 1 Feb 2016 20:28] Project setup, gitignore and blank readme (author: Alicia Sykes)
```

3.2.2 Git logs for main program

```
0a9c124 [Wed, 16 Mar 2016 12:25] New thumbnail for World Now page
(author: Alicia Sykes)
5958246 [Wed, 16 Mar 2016 12:24] updated placeholder text for search (author: Alicia Sykes)
8c1942c [Wed, 16 Mar 2016 12:23] Finished World Now page
(author: Alicia Sykes)
8c43b8b [Wed, 16 Mar 2016 12:23] Changed which pages show in the navbar (author: Alicia Sykes)
bd8dec0 [Wed, 16 Mar 2016 12:23] New thumbnail for World Now
(author: Alicia Sykes)
200551b [Wed, 16 Mar 2016 12:22] Added sentence and link about Twitter Now (author: Alicia
Sykes)
91025e4 [Wed, 16 Mar 2016 12:21] Updated conditions for saving tweets (author: Alicia Sykes)
6e6eb59 [Wed, 16 Mar 2016 12:21] Removed coverage badge, added link to hp entity-extraction
module (author: Alicia Sykes)
d12bbe [Wed, 16 Mar 2016 12:20] Fixed not showing older tweets, and refreshes automatically
(author: Alicia Sykes)
```

f7ccf95 [Wed, 16 Mar 2016 12:19] Remove oldest point when more that 400 reached, to stop browser crash (author: Alicia Sykes)

43929c5 [Wed, 16 Mar 2016 12:18] Implemented final changes to now main (author: Alicia Sykes)

68c96d1 [Wed, 16 Mar 2016 12:18] Open all external links in a new tab (author: Alicia Sykes)

c65a4b2 [Wed, 16 Mar 2016 10:12] Details about the MIT License (author: Alicia Sykes)

0dba42b [Sun, 13 Mar 2016 19:21] Finished displaying tweets live raw (author: Alicia Sykes)

6c7e984 [Sun, 13 Mar 2016 18:36] show loader while waiting for the map and other changes to now page (author: Alicia Sykes)

3bd863e [Sun, 13 Mar 2016 10:11] Real time tweet samples now work (author: Alicia Sykes)

360978a [Sun, 13 Mar 2016 09:30] Styles and layout for live bullet chart (author: Alicia Sykes)

4d213f6 [Sun, 13 Mar 2016 09:29] Finished real-time bullet chart (author: Alicia Sykes)

e1c8478 [Sun, 13 Mar 2016 09:28] Bullet.js dependency (author: Alicia Sykes)

05268d0 [Sun, 13 Mar 2016 07:58] now (author: Alicia Sykes)

3d3e092 [Sun, 13 Mar 2016 04:09] Browserified the now page (author: Alicia Sykes)

265178c [Sun, 13 Mar 2016 02:42] Now page has live map (author: Alicia Sykes)

400f4d3 [Sun, 13 Mar 2016 02:41] Modified stream handler to not save junk tweets with URLs (author: Alicia Sykes)

d90efa6 [Sun, 13 Mar 2016 01:49] Map is now reall time too - cool (author: Alicia Sykes)

3412cb6 [Sun, 13 Mar 2016 00:10] got static region map working (author: Alicia Sykes)

97dca3f [Sat, 12 Mar 2016 23:09] Implemened socket.io real-tim functionality into now page (author: Alicia Sykes)

293b6bf [Sat, 12 Mar 2016 22:20] Made static vertical bar chart (author: Alicia Sykes)

24fe3f5 [Sat, 12 Mar 2016 21:13] Setup Jade template for now page (author: Alicia Sykes)

871b58b [Sat, 12 Mar 2016 21:12] Modified save criteria (author: Alicia Sykes)

960472a [Sat, 12 Mar 2016 21:12] badges (author: Alicia Sykes)

91b0cc9 [Sat, 12 Mar 2016 20:23] badges (author: Alicia Sykes)

1c2388b [Sat, 12 Mar 2016 19:40] Created page and blank script for now page (author: Alicia Sykes)

679ce27 [Sat, 12 Mar 2016 19:40] Created new route for Now page (author: Alicia Sykes)

449461e [Sat, 12 Mar 2016 16:03] Badge (author: Alicia Sykes)

08681d1 [Sat, 12 Mar 2016 15:47] Added stackshare badge (author: Alicia Sykes)

2e4f784 [Sat, 12 Mar 2016 15:46] Updated rules for saving tweets (author: Alicia Sykes)

cf2900d [Sat, 12 Mar 2016 15:45] Now streams from the whole world (author: Alicia Sykes)

205a58e [Mon, 29 Feb 2016 15:38] New lines under title (author: Alicia Sykes)

2bc43df [Mon, 29 Feb 2016 15:35] Added user stories for the frontend of screens (author: Alicia Sykes)

0397475 [Mon, 29 Feb 2016 15:06] Wireframe page (author: Alicia Sykes)

7ff9207 [Mon, 29 Feb 2016 14:58] Frontend methodology (author: Alicia Sykes)

e0780fd [Mon, 29 Feb 2016 14:48] Added wireframes images (author: Alicia Sykes)

b65679a [Thu, 25 Feb 2016 18:13] Fixed minor typo in image path of screenshot documentation (author: Alicia Sykes)

b139144 [Thu, 25 Feb 2016 18:11] Added screenshots (author: Alicia Sykes)

18e4aa1 [Wed, 24 Feb 2016 18:42] Increased the efficiency of real-time scatter (author: Alicia Sykes)

24719c9 [Wed, 24 Feb 2016 18:42] Added mobile redirect (author: Alicia Sykes)

934e00e [Wed, 24 Feb 2016 18:41] Added mobile redirect (author: Alicia Sykes)

f6a8b46 [Wed, 24 Feb 2016 10:41] Fixed region map read url (author: Alicia Sykes)

c8670e4 [Wed, 24 Feb 2016 10:41] Fixed autocomplete url (author: Alicia Sykes)

f76baee [Wed, 24 Feb 2016 08:37] Modified socket.io to look on default port (author: Alicia Sykes)

d053c11 [Tue, 23 Feb 2016 16:53] Updated bower components (author: Alicia Sykes)

93f3122 [Tue, 23 Feb 2016 16:44] Commented out the real-time-dashboard route, which breaks everything (author: Alicia Sykes)

3f699d9 [Tue, 23 Feb 2016 16:43] So the real-time dashbaord works.... but, it crashes your whole computer and

s4f63dff [Tue, 23 Feb 2016 16:27] Improved colors, finished script (author: Alicia Sykes)

2ec2644 [Tue, 23 Feb 2016 15:33] Client side scatter working (author: Alicia Sykes)

824a760 [Mon, 22 Feb 2016 16:53] Created empty route for real-time dashboard (author: Alicia Sykes)

60bde46 [Mon, 22 Feb 2016 16:53] Improved styling and layout on home and search pages, and fixed search on sea7857576 [Mon, 22 Feb 2016 16:51] Show loading when navbar clicked (author: Alicia Sykes)

834b0fa [Mon, 22 Feb 2016 16:50] Added new thumbnails for latest data visualisations (author: Alicia Sykes)

1187f53 [Mon, 22 Feb 2016 14:22] Finished tone analysis front and backend (author: Alicia Sykes)

2469bad [Mon, 22 Feb 2016 14:22] Updated to work with the new tone analysis module (author: Alicia Sykes)

6507d74 [Mon, 22 Feb 2016 11:59] Browserified tones analyzer (author: Alicia Sykes)

6747ead [Mon, 22 Feb 2016 11:17] Error text (author: Alicia Sykes)

27c19a8 [Mon, 22 Feb 2016 10:29] Made tone pages (author: Alicia Sykes)

80db4b5 [Mon, 22 Feb 2016 09:18] Finished Trending sentiment page (author: **Alicia** Sykes)

cac5fed [Mon, 22 Feb 2016 09:18] Fixed missing D3 error (author: Alicia Sykes)

ee8ffa5 [Sun, 21 Feb 2016 13:07] Trending now full working with list of trends (author: Alicia Sykes)

ac9e59d [Sun, 21 Feb 2016 13:07] Updated fetch-tweets (author: Alicia Sykes)

db7e32d [Sun, 21 Feb 2016 10:58] Shows text trends from world-wide (author: Alicia Sykes)

d76333d [Sun, 21 Feb 2016 10:58] CSS loading spinner (author: Alicia Sykes)
0a15758 [Sun, 21 Feb 2016 10:21] Trending API done (author: Alicia Sykes)
15a9ddf [Sat, 20 Feb 2016 15:33] Added the trending route and files (author: Alicia Sykes)
30ad4b3 [Sat, 20 Feb 2016 14:55] Moved api routes into api folder (author: Alicia Sykes)
eefc87b [Fri, 19 Feb 2016 13:43] Implemented in page comparison to search (author: Alicia Sykes)
d18f823 [Fri, 19 Feb 2016 13:42] Added API route for db Tweets (author: Alicia Sykes)
338f232 [Fri, 19 Feb 2016 11:58] Finished implamenting entity analysis summary on search page (author: Alicia Sykes)
S527ab57 [Fri, 19 Feb 2016 11:58] Updated Materialize to latest version (author: Alicia Sykes)
91ef669 [Fri, 19 Feb 2016 10:58] Changed data structure and sorted numerically (author: Alicia Sykes)
9cf2c12 [Fri, 19 Feb 2016 10:26] Made a faster watch for coffeescript for dev (author: Alicia Sykes)
dc91aec [Fri, 19 Feb 2016 10:00] Backend working for entity extraction (author: Alicia Sykes)
c83e068 [Fri, 19 Feb 2016 09:59] Added route and code for entity-extraction api files (author: Alicia Sykes)
6210dc5 [Thu, 18 Feb 2016 12:00] Tooltips on bars (author: Alicia Sykes)
66dcd7b [Thu, 18 Feb 2016 11:47] Tone analysis using real data (author: Alicia Sykes)
68cea54 [Thu, 18 Feb 2016 11:40] Show loader while tones are loading (author: Alicia Sykes)
858753f [Thu, 18 Feb 2016 11:34] Front end for tone analysis done, changed layout of search page (author: Alicia Sykes)
7f4c420 [Wed, 17 Feb 2016 17:19] Files for tone analyzer (author: Alicia Sykes)
37d1258 [Wed, 17 Feb 2016 17:19] Added raw Tweets to search page and implemented radar (author: Alicia Sykes)
d38a319 [Wed, 17 Feb 2016 17:18] Basic styles for radar chart (author: Alicia Sykes)
d91cafe [Wed, 17 Feb 2016 17:14] Added new routes for tone analyzer (author: Alicia Sykes)
05d6ce0 [Wed, 17 Feb 2016 12:22] Implemented server side fetching tones from watson (author: Alicia Sykes)
048dfcb [Wed, 17 Feb 2016 11:36] Added buttons to globe (author: Alicia Sykes)
a0c1d83 [Wed, 17 Feb 2016 11:36] Show's top words used, into the search page (author: Alicia Sykes)
36a5fb2 [Mon, 15 Feb 2016 20:46] Fixed font size bug on word cloud (author: Alicia Sykes)
1c3371f [Mon, 15 Feb 2016 20:41] Round off percentages (author: Alicia Sykes)
51d7d6f [Mon, 15 Feb 2016 20:29] Ipdated fetch-tweets to fetch trends too (author: Alicia Sykes)
6faff0d [Mon, 15 Feb 2016 16:02] Not compatible with mobile message (author: Alicia Sykes)
d5dde51 [Mon, 15 Feb 2016 15:33] Increased font size (author: Alicia Sykes)
6be7464 [Mon, 15 Feb 2016 15:22] Updated the footer (author: Alicia Sykes)
5435314 [Mon, 15 Feb 2016 14:57] Created a side navbar for smaller screens, and made navbar responsive (author: Alicia Sykes)
db9d58f [Mon, 15 Feb 2016 13:30] Improved navbar styling (author: Alicia Sykes)
3cd71dd [Mon, 15 Feb 2016 12:53] Wrote about page (author: Alicia Sykes)
c4c83b7 [Mon, 15 Feb 2016 10:45] Updated phantom dev dependency (author: Alicia Sykes)
6ba7701 [Mon, 15 Feb 2016 10:41] Started the CSS refactoring marathon (author: Alicia Sykes)

2e0ce83 [Mon, 15 Feb 2016 10:41] Multipid size in bytes by 10 (author: Alicia Sykes)

b820da5 [Mon, 15 Feb 2016 09:51] Browserified bower components of scatter plot (author: Alicia Sykes)

b227b18 [Mon, 15 Feb 2016 09:45] Browsrified timelines bower components (author: Alicia Sykes)

8ae4830 [Mon, 15 Feb 2016 09:18] Made a custom error page (author: Alicia Sykes)

ec4a271 [Sun, 14 Feb 2016 16:51] Moved server-side sauce files (author: Alicia Sykes)

d710dee [Sun, 14 Feb 2016 16:19] Renamed source to client-side-souce (author: Alicia Sykes)

9524772 [Sun, 14 Feb 2016 11:47] Move url to seperate file (author: Alicia Sykes)

a5618da [Sun, 14 Feb 2016 11:34] Shortened app.js (author: Alicia Sykes)

9ad7cec [Sun, 14 Feb 2016 11:27] Removed keywords coloumn from db, and capped collection at 1500 (author: Alicia Sykes)

cbb8b2 [Sun, 14 Feb 2016 10:43] Added entry and thumbnail linhk for comparer into layout (author: Alicia Sykes)

1f9e57f [Sat, 13 Feb 2016 16:23] Slightly improved the search for brand comparison (author: Alicia Sykes)

bbd77dc [Sat, 13 Feb 2016 16:06] BASIC search form works for comparison page (author: Alicia Sykes)

c1d5f5f [Sat, 13 Feb 2016 15:24] Fix minor importing bug (author: Alicia Sykes)

b3a6a16 [Sat, 13 Feb 2016 15:20] Added links to tweet comparison (author: Alicia Sykes)

081fb75 [Sat, 13 Feb 2016 15:10] Shows list of top words used in Tweets (author: Alicia Sykes)

01d8efb [Sat, 13 Feb 2016 14:42] Shows overall sentiment (author: Alicia Sykes)

2501d83 [Sat, 13 Feb 2016 14:41] Get trending words (author: Alicia Sykes)

459908c [Sat, 13 Feb 2016 14:41] Modified to make code more reusable (author: Alicia Sykes)

bb87d6b [Sat, 13 Feb 2016 13:04] Working comparing Tweets (author: Alicia Sykes)

0a8bafd [Sat, 13 Feb 2016 10:47] Wrote server side code for fetching comparison Tweets (author: Alicia Sykes)

0847d7b [Sat, 13 Feb 2016 10:21] Created view, route and app.js entry for comparer (author: Alicia Sykes)

c94ae5a [Sat, 13 Feb 2016 10:20] Wrote script for asynchronously fetching tweets (author: Alicia Sykes)

ef17296 [Sat, 13 Feb 2016 08:25] Removed unused code from homepage (author: Alicia Sykes)

89860ab [Sat, 13 Feb 2016 07:50] Adjusted spacings and layout on home and search page (author: Alicia Sykes)

58f80f1 [Sat, 13 Feb 2016 07:26] Styled hexagons (author: Alicia Sykes)

b151869 [Sat, 13 Feb 2016 07:26] Database now succesfully capped at 1500 records (author: Alicia Sykes)

4db2a53 [Sat, 13 Feb 2016 07:25] Added GitHub and Read more button to home page (author: Alicia Sykes)

b5f6ede [Sat, 13 Feb 2016 05:47] REAL TIME HEXAGON TWEET BACKGROUND IS WORKING :) (author: Alicia Sykes)

a5fa8fb [Sat, 13 Feb 2016 00:23] Refracted js out of jade (author: Alicia Sykes)

1d94f05 [Fri, 12 Feb 2016 14:11] Modified home page layout and hexagons (author: Alicia Sykes)

94eaf44 [Fri, 12 Feb 2016 12:02] Modified homepage to display hexagons as background (author: Alicia Sykes)

f9d58d3 [Fri, 12 Feb 2016 12:01] Modified hexagon module to have different settings fo rhomepage (author: Alici4c49d73 [Fri, 12 Feb 2016 08:51] Update dependencies and fix build issue (author: Alicia Sykes)

18e494a [Wed, 10 Feb 2016 14:43] Modified fetch sentiment tweet object, made bigger hexagon diagram and added

2014388 [Wed, 10 Feb 2016 14:18] Added ripple effect to navbar, and tweaked search modules (author: Alicia Sykes)

1102dfb [Wed, 10 Feb 2016 13:40] Implemented enter to search on the homepage (author: Alicia Sykes)

8f9488e [Wed, 10 Feb 2016 12:06] Updated home page styles (author: Alicia Sykes)

43f05f9 [Wed, 10 Feb 2016 11:36] Created thumbnails for each vis (author: Alicia Sykes)

06a70fe [Wed, 10 Feb 2016 11:36] Modified image task to look in sub-directories (author: Alicia Sykes)

dec906d [Wed, 10 Feb 2016 10:32] Fixed height on time chart (author: Alicia Sykes)

ab53ec4 [Wed, 10 Feb 2016 10:30] Implemented gauge chart in search page (author: Alicia Sykes)

95bb2b3 [Wed, 10 Feb 2016 09:05] Changed search scripts to browserify bundle (author: Alicia Sykes)

6452efe [Wed, 10 Feb 2016 09:04] C3 dependency (author: Alicia Sykes)

42f10ee [Wed, 10 Feb 2016 08:49] Implemented tooltip functionality (author: Alicia Sykes)

9996670 [Wed, 10 Feb 2016 08:39] Basic hexbin vis working with real data (author: Alicia Sykes)

888f019 [Wed, 10 Feb 2016 08:39] Added dependency for d3 tip (author: Alicia Sykes)

61a8478 [Tue, 9 Feb 2016 15:35] Static hexbin working (author: Alicia Sykes)

65585c7 [Tue, 9 Feb 2016 15:12] Set up blank template for search page (author: Alicia Sykes)

57216ef [Tue, 9 Feb 2016 15:01] Little script to fetch Tweets and assign sentiment and keyword values (author: Alicia Sykes)

f7522b4 [Tue, 9 Feb 2016 14:41] Added new route, viwe and entry in app.js for the search page (author: Alicia Sykes)

3d40799 [Tue, 9 Feb 2016 14:26] Fixed moment depreication and refractored make-click-words into seperate file (author: Alicia Sykes)

678e528 [Tue, 9 Feb 2016 14:14] Added some blank tiles to the home page (author: Alicia Sykes)

e86762f [Tue, 9 Feb 2016 13:38] Deleted sample data (author: Alicia Sykes)

4c86004 [Tue, 9 Feb 2016 13:16] Finished entity extraction sankey chart (author: Alicia Sykes)

f8f8e10 [Tue, 9 Feb 2016 11:27] Implemented sample sankey chart with js and css (author: Alicia Sykes)

286357b [Tue, 9 Feb 2016 11:27] Added dependency for sankey d3 plugin (author: Alicia Sykes)

3ab6191 [Tue, 9 Feb 2016 10:48] Decorated the search box a bit (author: Alicia Sykes)

c61bd06 [Tue, 9 Feb 2016 10:39] Implemented press enter to search (author: Alicia Sykes)

f7c071a [Tue, 9 Feb 2016 10:28] Entity extraction frontend working, lots of jade loops (author: Alicia Sykes)

abd1066 [Tue, 9 Feb 2016 10:27] Entity extraction backend working (author: Alicia Sykes)

50d1679 [Tue, 9 Feb 2016 10:27] Styles for entity extraction objects (author: Alicia Sykes)

87e7a16 [Tue, 9 Feb 2016 10:27] Modified the card shape to have margin and no padding (author: Alicia Sykes)

4860916 [Mon, 8 Feb 2016 17:25] Created route, fetch tweets and extract etities then calls Jade layout (author: Alicia Sykes)

f31db7e [Mon, 8 Feb 2016 17:25] Set up view and script for entity-extraction (author: Alicia Sykes)

50d95e0 [Mon, 8 Feb 2016 17:24] Installed HP entity extraction module (author: Alicia Sykes)

ea6931c [Mon, 8 Feb 2016 17:24] Created route for entity-extraction (author: Alicia Sykes)

2a74150 [Mon, 8 Feb 2016 14:20] Got rid of break-down tweets from the navbar, cos it's rubbish - relaced it (author: Alicia Sykes)

295a6f1 [Mon, 8 Feb 2016 14:16] Tried (and mostly failed) to fix that large font size bug (author: Alicia Sykes)

dea6116 [Mon, 8 Feb 2016 14:05] Finished Tweet break-down chart, implemented functionality and styled (author: Alicia Sykes)

772f552 [Mon, 8 Feb 2016 12:57] Now works for database results too (author: Alicia Sykes)

73e9991 [Mon, 8 Feb 2016 12:27] Break down workingfor fresh Tweets basic first version (author: Alicia Sykes)

05496e6 [Sun, 7 Feb 2016 17:36] Styled and modified layout ready for real data in a minute (author: Alicia Sykes)

0db01e7 [Sun, 7 Feb 2016 12:09] Created route, view and script for tweet tree map (author: Alicia Sykes)

35e85af [Sun, 7 Feb 2016 12:09] New styles for the break-down tree map (author: Alicia Sykes)

3115362 [Sun, 7 Feb 2016 12:08] Added new entry for Tweet-break-down (author: Alicia Sykes)

180aeca [Sun, 7 Feb 2016 12:08] Added new route for Tweet break-down (author: Alicia Sykes)

b7116be [Sat, 6 Feb 2016 16:44] Show moment.js time for each incoming Tweet (author: Alicia Sykes)

6b4a065 [Sat, 6 Feb 2016 15:56] Press enter to search (author: Alicia Sykes)

34eafef [Sat, 6 Feb 2016 15:48] Filter live tweets LIVE, very cool (author: Alicia Sykes)

58ec391 [Sat, 6 Feb 2016 15:48] Implemented search term feature (author: Alicia Sykes)

7953046 [Sat, 6 Feb 2016 15:08] Went back to non-browserified script (author: Alicia Sykes)

d9f8f62 [Sat, 6 Feb 2016 14:53] Added switch to turn real-time on and off and changed script path to bundle (author: Alicia Sykes)

54bb68c [Sat, 6 Feb 2016 14:30] Displays real-time Tweets in real-time, awesome (author: Alicia Sykes)

c801d08 [Sat, 6 Feb 2016 14:30] Modified stram handler to also emit anyTweet without location (author: Alicia Sykes)

602449c [Sat, 6 Feb 2016 13:24] Moified sorting of Tweets, and added search box (author: Alicia Sykes)

5fc3c12 [Sat, 6 Feb 2016 12:33] Showing database tweets (author: Alicia Sykes)

0e337f3 [Sat, 6 Feb 2016 12:33] Added additional text styles (author: Alicia Sykes)

9e243ca [Sat, 6 Feb 2016 10:49] Added new route, page and navbar entry for the raw tweets section (author: Alicia Sykes)

7b60ca6 [Fri, 5 Feb 2016 15:26] Now shows statistics on region map, finished (author: Alicia Sykes)

c11ea68 [Fri, 5 Feb 2016 15:26] Finished region map backend, better sorted data (author: Alicia Sykes)

ac3ec1d [Fri, 5 Feb 2016 15:25] Added font colours into percentages (author: Alicia Sykes)

ce965e3 [Fri, 5 Feb 2016 12:24] Places autocomplete now uses live data (author: Alicia Sykes)

159c905 [Fri, 5 Feb 2016 11:50] Static autocomplete implemented (author: Alicia Sykes)

ae06338 [Fri, 5 Feb 2016 11:35] Now results are better (author: Alicia Sykes)

6083f2e [Wed, 3 Feb 2016 12:25] Modified map colours (author: Alicia Sykes)

0779eb7 [Wed, 3 Feb 2016 11:53] Calculates region from lat and long for map, and sends live data (author: Alicia Sykes)

f689eb6 [Wed, 3 Feb 2016 11:52] Reversed the sentiment color, and now uses live data (author: Alicia Sykes)

ceb764f [Wed, 3 Feb 2016 11:52] Added dependency for my new region finding module (author: Alicia Sykes)

401c3b7 [Wed, 3 Feb 2016 11:16] Added module links to radme (author: Alicia Sykes)

b9d8cf9 [Mon, 1 Feb 2016 14:55] Created and configured reoutes and files for region map (author: Alicia Sykes)

e0f62ea [Mon, 1 Feb 2016 14:54] Added region map to nav bar (author: Alicia Sykes)

b17318e [Mon, 1 Feb 2016 14:15] Searching on the timeline now works, finished (author: Alicia Sykes)

b272d51 [Mon, 1 Feb 2016 14:15] Fixed forceX bug (author: Alicia Sykes)

de9edfb [Sat, 30 Jan 2016 15:58] Display first version of sentiment timeline working (author: Alicia Sykes)

b31a582 [Sat, 30 Jan 2016 15:57] Timeline route and query working (author: Alicia Sykes)

0c31ef9 [Sat, 30 Jan 2016 15:56] D3 code for sentiment timeline done (author: Alicia Sykes)

e4373c6 [Sat, 30 Jan 2016 15:55] Script to fetch and format sentiment data for the timeline, working nicley (aua3f7e41 [Fri, 29 Jan 2016 16:55] Set up basics for area line chart (author: Alicia Sykes)

f767d71 [Fri, 29 Jan 2016 16:54] Full width adjustment (author: Alicia Sykes)

fa128c7 [Fri, 29 Jan 2016 16:54] Added dependency for NVD3 (author: Alicia Sykes)

b9f81d8 [Fri, 29 Jan 2016 13:12] Retains search term when switching between word cloud and scatter plot (author:b550e59 [Fri, 29 Jan 2016 13:11] Scatter plot now uses real data (author: Alicia Sykes)

bc12c4e [Fri, 29 Jan 2016 13:10] Show loader when search or enter is pressed (author: Alicia Sykes)

7b04711 [Fri, 29 Jan 2016 13:10] Finished scatter plot (author: Alicia Sykes)

06fe13a [Fri, 29 Jan 2016 12:03] Almost finished D3 scatter (author: Alicia Sykes)

405fb69 [Fri, 29 Jan 2016 10:54] git (author: Alicia Sykes)

3820150 [Fri, 29 Jan 2016 10:49] Positioned Scatter (author: Alicia Sykes)

751d862 [Fri, 29 Jan 2016 10:23] Updated scattr plot position (author: Alicia Sykes)

da074b8 [Fri, 29 Jan 2016 10:17] Removed obsolete CSS and positioned Scattre Plot (author: Alicia Sykes)

e97e1cd [Fri, 29 Jan 2016 10:16] Refracted the D3 CoffeeScript (author: Alicia Sykes)

c3ab03a [Thu, 28 Jan 2016 09:00] Basic Scatter Plot (author: Alicia Sykes)

7d92447 [Thu, 28 Jan 2016 09:00] Basic Scatter Plot (author: Alicia Sykes)

df73349 [Thu, 28 Jan 2016 16:20] Added route to word plot (author: Alicia Sykes)

4b958c4 [Thu, 28 Jan 2016 16:19] Fixed link to word plot (author: Alicia Sykes)

4c13d2f [Thu, 28 Jan 2016 16:01] Removed that cheeky console.log which snuck through (author: Alicia Sykes)

bea94e6 [Thu, 28 Jan 2016 16:01] Word cloud done (author: Alicia Sykes)

5313eaf [Thu, 28 Jan 2016 16:00] Tabs and links on completed word cloud page (author: Alicia Sykes)

1a86cc9 [Thu, 28 Jan 2016 15:04] Calculates and displays trending keywords in latest Tweets (author: Alicia Syk69fcf75 [Thu, 28 Jan 2016 14:47] Now shows top positive and negative words with REAL data (author: Alicia Sykes2982d94 [Thu, 28 Jan 2016 14:03] Formated the top words on word cloud (author: Alicia Sykes)

fddeab4 [Thu, 28 Jan 2016 13:39] Path to pass top positive and negative words (author: Alicia Sykes)

21f449a [Thu, 28 Jan 2016 09:13] Can now search word cloud from page controls (author: Alicia Sykes)

e63d48e [Thu, 28 Jan 2016 09:13] Made page controls generic and reusable (author: Alicia Sykes)

fb21038 [Thu, 28 Jan 2016 08:46] Adjusted size of words (author: Alicia Sykes)

017cf48 [Wed, 27 Jan 2016 13:30] Started making controls for word cloud (author: Alicia Sykes)

67a9591 [Wed, 27 Jan 2016 13:30] Modified CSS for controls (author: Alicia Sykes)

3412065 [Wed, 27 Jan 2016 11:30] User can now click on a word in word cloud (author: Alicia Sykes)

a340a88 [Wed, 27 Jan 2016 11:21] Added a route for custom words (author: Alicia Sykes)

fb38527 [Wed, 27 Jan 2016 10:39] Sizing and positioning of word cloud (author: Alicia Sykes)

14437e4 [Tue, 26 Jan 2016 16:55] Sizing of text now works (author: Alicia Sykes)

650233c [Tue, 26 Jan 2016 16:54] Updated to the latest version of remove-words (author: Alicia Sykes)

7f36ec3 [Tue, 26 Jan 2016 14:22] Basic word cloud now reads from db (author: Alicia Sykes)

b23808a [Tue, 26 Jan 2016 13:49] Set word cloud to get data from server side (author: Alicia Sykes)

b564c6d [Mon, 25 Jan 2016 17:32] Modified chart size and opacity (author: Alicia Sykes)

3d7d47d [Mon, 25 Jan 2016 17:20] Rotated text by random multiple of 45 (author: Alicia Sykes)

726e83c [Mon, 25 Jan 2016 17:06] Defines SVG in Jade template and kind of sorted that size issue (author: Alicibba257e [Mon, 25 Jan 2016 16:24] Implemented size scale (author: Alicia Sykes)

02fe524 [Mon, 25 Jan 2016 14:02] Fixed error in scale colors (author: Alicia Sykes)

0b9e615 [Mon, 25 Jan 2016 13:29] Red - Green scale for words (author: Alicia Sykes)

36ba39b [Mon, 25 Jan 2016 13:29] Added simple D3 static word cloud in (author: Alicia Sykes)

b03489f [Mon, 25 Jan 2016 10:23] Added word cloud into navbar (author: Alicia Sykes)

1288656 [Mon, 25 Jan 2016 10:22] Created route for word cloud page (author: Alicia Sykes)

bc973fb [Wed, 20 Jan 2016 11:01] Toggle heat map implemented (author: Alicia Sykes)

942bfd8 [Wed, 20 Jan 2016 10:53] Styles for clear map btn (author: Alicia Sykes)

93b4a45 [Wed, 20 Jan 2016 10:48] Filter positive and negative maps (author: Alicia Sykes)

b82a2df [Mon, 18 Jan 2016 16:00] Applied opacity value to heatmap layer (author: Alicia Sykes)

af1807e [Mon, 18 Jan 2016 15:47] Number of Tweets counter increments in real-time (author: Alicia Sykes)

eec16d6 [Mon, 18 Jan 2016 15:27] Made summary text for globe (author: Alicia Sykes)

de7bed2 [Mon, 18 Jan 2016 12:51] Implemented Coffee Lint code quality suggestions (author: Alicia Sykes)

039eb34 [Mon, 18 Jan 2016 12:38] All dependencies up to date and removed JSX (author: Alicia Sykes)

6ff00bd [Mon, 18 Jan 2016 12:37] Removed JSX reference (author: Alicia Sykes)

315b0f5 [Mon, 18 Jan 2016 12:37] Updated to work with latest version of del (author: Alicia Sykes)

050a5dd [Mon, 18 Jan 2016 11:51] Updated dev dependencies (author: Alicia Sykes)

c0de153 [Sun, 17 Jan 2016 16:25] Globe is now realtime (author: Alicia Sykes)

4fdb1aa [Sun, 17 Jan 2016 16:20] Added add data method in globe bundle (author: Alicia Sykes)

1d5a4fb [Sun, 17 Jan 2016 11:01] Only show live results on full map (author: Alicia Sykes)

f72b206 [Sun, 17 Jan 2016 10:50] Refracted socket into map bundle (author: Alicia Sykes)

fa7f507 [Sun, 17 Jan 2016 10:41] If searched then show search term and hide clear btn (author: Alicia Sykes)

8be7369 [Sat, 16 Jan 2016 12:22] Deleted the old live-map page and made the non-live map live (author: Alicia S90e35fb [Sat, 16 Jan 2016 12:10] Fixed incorrect method name err (author: Alicia Sykes)

c726fe2 [Sat, 16 Jan 2016 11:53] Refracted live interactions into separate file (author: Alicia Sykes)

86a83c5 [Sat, 16 Jan 2016 11:26] Improved real-time map and added clear btn (author: Alicia Sykes)

1186f4f [Fri, 15 Jan 2016 13:35] Real time map working (author: Alicia Sykes)

7c51835 [Fri, 15 Jan 2016 13:35] GOT LIVE MAP WORKING (author: Alicia Sykes)

48ffb05 [Fri, 15 Jan 2016 13:34] Remove unnecessary code (author: Alicia Sykes)

20e804c [Fri, 15 Jan 2016 13:34] Added new frontend dependency for socket.io (author: Alicia Sykes)

c9c842b [Fri, 15 Jan 2016 13:33] Now streams A LOT of Tweets FAST (author: Alicia Sykes)

3014147 [Wed, 13 Jan 2016 13:06] Removed commented out code (author: Alicia Sykes)

c421d6d [Wed, 13 Jan 2016 13:04] Moved streaming into app.js and fixed dupp bug (author: Alicia Sykes)

8a358bd [Wed, 13 Jan 2016 12:03] Refracted globe bundle into separate files (author: Alicia Sykes)

bee44ad [Wed, 13 Jan 2016 12:03] Added debrowserify to scripts gulp task to bundle bower_components (author: Al259084e [Wed, 13 Jan 2016 10:59] Shows search term in textbox (author: Alicia Sykes)

3367767 [Tue, 12 Jan 2016 18:00] Show and hide space (author: Alicia Sykes)

b423c6e [Tue, 12 Jan 2016 17:38] Smallened the info window on globe (author: Alicia Sykes)

71ee6ea [Tue, 12 Jan 2016 16:27] Show and hide details (author: Alicia Sykes)

aaa2165 [Tue, 12 Jan 2016 15:03] Made search box and summary for globe page (author: Alicia Sykes)

109545e [Tue, 12 Jan 2016 12:59] Modified colors of world image for the globe (author: Alicia Sykes)

798d9d3 [Tue, 12 Jan 2016 12:59] Added globe link to the navbar (author: Alicia Sykes)

5082c7f [Tue, 12 Jan 2016 12:36] Option to hide nav (author: Alicia Sykes)

d80a9fd [Mon, 11 Jan 2016 15:41] Refracted and all working (author: Alicia Sykes)

aab1e68 [Mon, 11 Jan 2016 14:59] Showing real lat + lng on globe (author: Alicia Sykes)

56029a9 [Mon, 11 Jan 2016 14:23] Globe fetches real data (but does nothing with it yet) (author: Alicia Sykes) e816340 [Mon, 11 Jan 2016 14:17] Uses correct colours for positive and negative sentiment (author: Alicia Sykesaed68bb [Sun, 10 Jan 2016 18:09] Nice picture of the world for the webgl globe (author: Alicia Sykes)

69dae4b [Sun, 10 Jan 2016 18:09] Changed globe scripts to be a browserify bundle (author: Alicia Sykes)

a53b733 [Sun, 10 Jan 2016 18:07] Modified images task to accept jpegs too (author: Alicia Sykes)

ad2210e [Sat, 9 Jan 2016 22:08] Created skeleton for sentiment globe (author: Alicia Sykes)

27e5838 [Fri, 8 Jan 2016 18:00] Added formatting to the map summary text (author: Alicia Sykes)

ca672df [Fri, 8 Jan 2016 15:42] Added a neutral gradient for results with zero sentiment (author: Alicia Sykes)b8fffa0 [Fri, 8 Jan 2016 15:07] Submit search term when the user presses enter (author: Alicia Sykes)

01c0176 [Fri, 8 Jan 2016 14:29] Refracted the summary sentences into separate class (author: Alicia Sykes)

887ad6a [Fri, 8 Jan 2016 13:57] Show basic stats for map in footer (author: Alicia Sykes)

4a6f57e [Thu, 7 Jan 2016 13:59] Made params for sending summary data to map (author: Alicia Sykes)

f9de133 [Thu, 7 Jan 2016 12:11] Updated remove-words to latest version, better words list (author: Alicia Sykes)89b52ce [Thu, 7 Jan 2016 12:01] Template footer for bottom of map page (author: Alicia Sykes)

c4a0999 [Thu, 7 Jan 2016 12:00] Made a key for the sentiment heat map (author: Alicia Sykes)

2bbc8cf [Wed, 6 Jan 2016 15:26] Developed new style theme for map (author: Alicia Sykes)

5677125 [Wed, 6 Jan 2016 14:40] Improved and finalised detailed heatmap gradient (author: Alicia Sykes)

816e9e1 [Wed, 6 Jan 2016 13:18] Refracted and neatened heatmap module slightly (author: Alicia Sykes)

b96bfa6 [Wed, 6 Jan 2016 13:02] Improved heatmap colours (author: Alicia Sykes)

eb29f29 [Wed, 6 Jan 2016 11:57] hmm not sure why this file is untracked.... (author: Alicia Sykes)

9c2c030 [Wed, 6 Jan 2016 11:57] Finished popup box for markers (author: Alicia Sykes)

004173b [Tue, 5 Jan 2016 14:28] Nicer custom content window for the markers on the map shows the sentiment (author: Alicia Sykes)

5bfcc15 [Tue, 5 Jan 2016 09:33] Modified max and default zoom to show whole map by default (author: Alicia Sykes)

6291ad1 [Mon, 4 Jan 2016 12:28] Database is now capped to 100 mb (author: Alicia Sykes)

b22f6fe [Mon, 4 Jan 2016 12:07] Only insert Tweet if it doesnt yet exist (author: Alicia Sykes)

c125e67 [Mon, 4 Jan 2016 10:45] Refracted map route into more testable code (author: Alicia Sykes)

91288fe [Mon, 4 Jan 2016 10:33] Refractor the map logic out of the map route (author: Alicia Sykes)

08912cc [Mon, 4 Jan 2016 09:51] Now includes database results on heatmap (author: Alicia Sykes)

a9f0255 [Sun, 3 Jan 2016 11:17] Update place-lookup, to fix not loading error (author: Alicia Sykes)

a5116be [Thu, 31 Dec 2015 14:11] Slightly blur location data, as to not reveal users exact position (author: Alca644af [Wed, 30 Dec 2015 16:52] Refracted the marker cluster down into separate class (author: Alicia Sykes)5824c40 [Wed, 30 Dec 2015 15:20] Removed google places from map (author: Alicia Sykes)

2015425 [Wed, 30 Dec 2015 15:16] Hide cluster text (author: Alicia Sykes)

0ea22c5 [Wed, 30 Dec 2015 14:47] Transparent cluster map V0.1 (author: Alicia Sykes)

389a1b0 [Wed, 30 Dec 2015 14:47] Nice picture of nothing for transparent marker (author: Alicia Sykes)

7d32a72 [Wed, 30 Dec 2015 14:46] Package for Google cluster markers (author: Alicia Sykes)

bb5fd13 [Tue, 29 Dec 2015 16:35] Loading (author: Alicia Sykes)

f382cb4 [Tue, 29 Dec 2015 16:35] comparison of dffernt sa methods jade (author: Alicia Sykes)

936fc50 [Tue, 29 Dec 2015 16:34] Bit closer to real-time (author: Alicia Sykes)

d24b8b2 [Tue, 29 Dec 2015 16:28] Changed gradiets for heatmap (author: Alicia Sykes)

5266eec [Sun, 27 Dec 2015 21:21] Documented the SA Comparison script (author: Alicia Sykes)

0dd11ff [Fri, 25 Dec 2015 20:44] Attaches keywords from tweets to oject (author: Alicia Sykes)

a4ba3c9 [Fri, 25 Dec 2015 20:43] Displays sentiment data from db (author: Alicia Sykes)

bf2e718 [Fri, 25 Dec 2015 20:41] Modified Tweet object to return all data (author: Alicia Sykes)

aaf6f60 [Sun, 20 Dec 2015 21:25] Fixed google places api key bug, working (author: Alicia Sykes)

5815d5f [Sun, 20 Dec 2015 21:24] Now working shows actual real sentiment data (author: Alicia Sykes)

3ecc62c [Sun, 20 Dec 2015 21:24] Updates place-lookup to latest (author: Alicia Sykes)

137cfe9 [Mon, 14 Dec 2015 19:15] Wrote a very nice class to handle the asynchronus fetching of Tweets, locations048f960 [Tue, 24 Nov 2015 18:53] Heat map data now passed from server side to client (author: Alicia Sykes)

649c6b1 [Tue, 24 Nov 2015 17:17] Neatened code (author: Alicia Sykes)

64f6d5f [Sun, 22 Nov 2015 17:18] bigger font, better labels (author: Alicia Sykes)

d0a19f1 [Sun, 22 Nov 2015 17:17] Override butotn colours (author: Alicia Sykes)

bfeba18 [Sun, 22 Nov 2015 17:15] Added spacing to page diivider (author: Alicia Sykes)

565472f [Sun, 22 Nov 2015 17:15] Modified zindex of loading graphic (author: Alicia Sykes)

7243efc [Sun, 22 Nov 2015 17:14] Fixed footer padding and added white bg class (author: Alicia Sykes)

c4be84a [Sun, 22 Nov 2015 17:10] Changed base colours (author: Alicia Sykes)

ed36cae [Sun, 22 Nov 2015 17:09] Create html table manually instead of Google Charts (author: Alicia Sykes)

f1c5e6a [Sun, 22 Nov 2015 17:09] Added function to display loading graphic (author: Alicia Sykes)

f4e27ab [Sun, 22 Nov 2015 17:08] Increased font size in radar chart graphic (author: Alicia Sykes)

0522326 [Sun, 22 Nov 2015 17:07] Removed redundant bower dependencies (author: Alicia Sykes)

b75dc49 [Sun, 22 Nov 2015 12:42] Better range for summary chart (author: Alicia Sykes)

4253367 [Fri, 20 Nov 2015 08:57] Added summary chart and radar diagram (author: Alicia Sykes)

dd83487 [Thu, 12 Nov 2015 14:33] Shows detailed sentiment comparison visualisations for any query (author: Alicb8735a5 [Wed, 11 Nov 2015 15:10] Set new gulp scripts location (author: Alicia Sykes)

af77f11 [Wed, 11 Nov 2015 15:09] Now shows simple scatter graph with results (author: Alicia Sykes)

9b8f519 [Tue, 10 Nov 2015 13:47] Wrote base code for calculating and comparing SA methods (author: Alicia Sykes)

fc5cf76 [Tue, 10 Nov 2015 13:46] part of public dir unignored and removed from clean task (author: Alicia Sykes)

4ad5faf [Mon, 19 Oct 2015 10:33] Now successfullly gets place data for nearly all Tweets and emmits (author: Alicia Sykes)

06827d6 [Thu, 15 Oct 2015 13:30] Included AFINN-1111 sentiment analysis and remove-words for streamed Tweet model (author: Alicia Sykes)

9eed8d8 [Wed, 14 Oct 2015 21:44] Link to new hp-sentiment module (author: Alicia Sykes)

6624965 [Thu, 1 Oct 2015 13:14] Added module links (author: Alicia Sykes)

0012798 [Thu, 24 Sep 2015 20:33] Added test command (author: Alicia Sykes)

a4ea6e5 [Thu, 24 Sep 2015 20:19] Corrected url for test-streategy in contents (author: Alicia Sykes)

de5ce0b [Thu, 24 Sep 2015 20:16] Contents page for documentation (author: Alicia Sykes)

ba418ce [Thu, 24 Sep 2015 20:10] Added Phantomjs into the test strategy (author: Alicia Sykes)

299cef5 [Thu, 24 Sep 2015 20:06] Added summary of test strategy to readme (author: Alicia Sykes)

e72b8ad [Thu, 24 Sep 2015 19:41] Wrote test strategy (author: Alicia Sykes)

2f76d26 [Tue, 22 Sep 2015 20:20] Created a config file to avoid strings in app.js (author: Alicia Sykes)

e57504d [Tue, 22 Sep 2015 20:00] Commented out temporarily not needed line (author: Alicia Sykes)

585e492 [Tue, 22 Sep 2015 19:59] Remove redundant return statment (author: Alicia Sykes)

e1f0448 [Tue, 22 Sep 2015 19:58] Changed a few words (author: Alicia Sykes)

4656392 [Tue, 22 Sep 2015 19:58] Added gulp-recess (author: Alicia Sykes)

bb7900e [Tue, 22 Sep 2015 19:21] Wrote documentation for build process (author: Alicia Sykes)

f620ba5 [Tue, 22 Sep 2015 19:20] Wrote documentation for build process (author: Alicia Sykes)

620390c [Tue, 22 Sep 2015 14:18] Added user stories, yaay user stories (author: Alicia Sykes)

b740d3d [Tue, 22 Sep 2015 13:56] Removed Socket integration from app.js (author: Alicia Sykes)

dcc2d0e [Tue, 22 Sep 2015 12:59] Added link to Trello board (author: Alicia Sykes)

50c0939 [Mon, 21 Sep 2015 22:56] Updated badges (author: Alicia Sykes)

113465c [Mon, 21 Sep 2015 22:54] Removed unused vars (author: Alicia Sykes)

792b1c4 [Mon, 21 Sep 2015 22:51] Applied the advice from the linter to the CSS and Less files (author: Alicia Sykes)

a32820e [Mon, 21 Sep 2015 22:01] Removed unused variables and requires (author: Alicia Sykes)

fbecfb9 [Mon, 21 Sep 2015 21:54] Changed dev mode flag to true by default (author: Alicia Sykes)

b05d335 [Mon, 21 Sep 2015 21:53] Fixed issues picked up by CoffeeLint (author: Alicia Sykes)

3348860 [Mon, 21 Sep 2015 21:37] Removed unused require (author: Alicia Sykes)

7b4efad [Mon, 21 Sep 2015 21:34] Updated dependncies (author: Alicia Sykes)

13e2dc0 [Mon, 21 Sep 2015 20:32] Chnaged test script (author: Alicia Sykes)

58917eb [Mon, 21 Sep 2015 20:28] Updated Ttavis config so he will stop emailing me everytime I make a commit (a7034998 [Mon, 21 Sep 2015 20:10] Coreected small typo in installation instrucitons (author: Alicia Sykes)

82eaa5f [Mon, 21 Sep 2015 20:05] Wrote guide on running locally and moved out of readme (author: Alicia Sykes)

e600ccf [Mon, 21 Sep 2015 13:30] Put some nice badges on the reame, which seem to be indicating that my project

a955a5 [Mon, 21 Sep 2015 13:29] Added Travis configuration file, for CI testing (author: Alicia Sykes)

563f50b [Sun, 20 Sep 2015 19:48] Added --dev flag option for all gulp tasks to increase awesomness (author: Alicia Sykes)

29a8075 [Sun, 20 Sep 2015 14:17] Don't include public directory as it is generated (author: Alicia Sykes)

09adfcf [Sun, 20 Sep 2015 14:15] Actually nothing [changed here - git your stupid (author: Alicia Sykes)

80e1680 [Sun, 20 Sep 2015 14:14] Improved the logic flow of the gulp process for speed (author: Alicia Sykes)

0432192 [Sun, 20 Sep 2015 07:43] Corrected typo in install instructions (author: Alicia Sykes)

588d8b6 [Sun, 20 Sep 2015 07:39] Now uses stream-tweets node module, rather than deprecated nTwitter (author: Alicia Sykes)

73a5bbc [Sun, 20 Sep 2015 07:38] Wrote how to run locally instructions (author: Alicia Sykes)

ef47632 [Sun, 20 Sep 2015 07:36] Code for generating new twitter keys file if not exist (author: Alicia Sykes) **68f645f** [Sat, 19 Sep 2015 17:03] Wrote installation instructions (author: Alicia Sykes)

cf85450 [Sat, 19 Sep 2015 15:14] Added loading gif to source (author: Alicia Sykes)

3e45ea2 [Sat, 19 Sep 2015 15:14] Ignore npm debug logs if they're there (author: Alicia Sykes)

b42b3be [Sat, 19 Sep 2015 15:13] Change autocomplete colours (author: Alicia Sykes)

11bd99c [Sat, 19 Sep 2015 15:10] Added new dependencies for full gulp build process (author: Alicia Sykes)

967330b [Sat, 19 Sep 2015 15:10] Gulp tasks to build all source files now included (author: Alicia Sykes)

68b8136 [Sat, 19 Sep 2015 15:08] Ignore built files (author: Alicia Sykes)

30ff5b0 [Sat, 19 Sep 2015 15:06] Added CoffeeScript routes (author: Alicia Sykes)

efe18bc [Sat, 19 Sep 2015 15:02] Removed all built files (author: Alicia Sykes)

833e49c [Fri, 18 Sep 2015 19:40] Implemented live streaming Tweets with realtime awesomness (author: Alicia Sykes)

7ad386f [Fri, 18 Sep 2015 19:39] Mongoose code for handling the Tweets (author: Alicia Sykes)

070f80f [Fri, 18 Sep 2015 19:37] Wrote react.js code for updating Tweets live (author: Alicia Sykes)

2f86a17 [Fri, 18 Sep 2015 19:33] Connect to Mongodb and Socket.io TODO: refactor (author: Alicia Sykes)

8777c53 [Fri, 18 Sep 2015 19:07] Installed react.js dependencies (author: Alicia Sykes)

1f5b8f6 [Fri, 18 Sep 2015 19:06] Don't push API keys to git (author: Alicia Sykes)

e526510 [Fri, 18 Sep 2015 19:04] Added more script paths, no longer uglifys js (author: Alicia Sykes)

e212e89 [Fri, 18 Sep 2015 19:03] Modified b-sync task to be less annoying (author: Alicia Sykes)

37c555b [Fri, 18 Sep 2015 19:02] Now includes reactify for jsx browserifying as well (author: Alicia Sykes)

506ebfa [Wed, 9 Sep 2015 12:29] Modified gulp scripts to run on multiple sources (author: Alicia Sykes)

9d1d76e [Wed, 9 Sep 2015 11:04] Refactored app.js and connected to MongoDB (author: Alicia Sykes)

98a776e [Wed, 9 Sep 2015 10:06] Added route and page for new live map (author: Alicia Sykes)

efba2ad [Tue, 1 Sep 2015 21:27] Added missing dependencies (author: Alicia Sykes)

d02f91e [Tue, 25 Aug 2015 17:48] Coverage reports now save to reports directory (author: alicia.sykes)

a396aae [Thu, 20 Aug 2015 17:56] Implemented basic autocomplete for map keywords (author: alicia.sykes)

39e276d [Sun, 16 Aug 2015 19:56] Neatened up the new map code (author: Alicia Sykes)

ccedb21 [Sun, 16 Aug 2015 19:39] Refactored the map script into modules for Browserify (author: Alicia Sykes)

9a05977 [Sat, 15 Aug 2015 19:47] Pushing compiled JavaScript, from new Gulp process (author: Alicia Sykes)

b5e7544 [Sat, 15 Aug 2015 19:43] Split browserify and script watch tasks for speed (author: Alicia Sykes)

fcc924d [Sat, 15 Aug 2015 19:22] Implemented the browserify task into gulp default process (author: Alicia Sykes)

6bf1eea [Sat, 15 Aug 2015 18:30] Improved scripts task, now using gulp-filter instead of event-stream (author: Alicia Sykes)

71f088e [Sat, 15 Aug 2015 16:49] Created browserify gulp task (author: Alicia Sykes)

03c14fb [Sun, 9 Aug 2015 14:06] Multiple heat map layers displayed on map (author: Alicia Sykes)

f4a6fa0 [Sat, 8 Aug 2015 18:26] Modified the Google configuration of the map (author: Alicia Sykes)

fa338ff [Fri, 7 Aug 2015 20:59] Slightly neatened the map code and made a start on the heat map (author: Alicia Sykes)

d655bb3 [Fri, 7 Aug 2015 06:30] Converted map scripts to CoffeScript (author: Alicia Sykes)

1e299a0 [Fri, 7 Aug 2015 06:18] Fiexed bsync is not defined in timeout errorr (author: Alicia Sykes)

69e87f6 [Thu, 6 Aug 2015 21:28] Styles the map with basic params (author: Alicia Sykes)

9b6da84 [Thu, 6 Aug 2015 19:12] Implemented places autocomplete search and integrated with map (author: Alicia 2cf5075 [Thu, 6 Aug 2015 14:56] Added menu bar to map screen (author: Alicia Sykes)

9597f4b [Thu, 6 Aug 2015 14:51] JS is now configured to automatically minify in gulp (author: Alicia Sykes)

c9f201a [Wed, 5 Aug 2015 14:28] Added mocha tests and coverage into gulp process (author: Alicia Sykes)

18459d7 [Tue, 4 Aug 2015 14:26] Refracted the gulpfile down into individual task files (author: Alicia Sykes)

7dbc763 [Tue, 4 Aug 2015 07:09] Added a test method into gulp file (author: Alicia Sykes)

21b29fd [Mon, 3 Aug 2015 20:04] Added mochaawesome to create reports - thanks adamgruber (author: Alicia Sykes)

fb63881 [Mon, 3 Aug 2015 17:35] Started implementing test structure (author: Alicia Sykes)

8edcebd [Sun, 2 Aug 2015 08:44] Created readme (author: Alicia Sykes)

0af4e0f [Sat, 1 Aug 2015 19:31] Added empty about page for later (author: Alicia Sykes)

7b7ea77 [Sat, 1 Aug 2015 13:16] Applied new UI changes (author: Alicia Sykes)

fb3f2b4 [Fri, 31 Jul 2015 22:27] Added dependancy for browser-sync (author: Alicia Sykes)

4f19faf [Fri, 31 Jul 2015 21:57] Created initial project structure (author: **Alicia** Sykes)

4bef537 [Fri, 31 Jul 2015 21:54] Created initial project config and dependencies files (author: Alicia Sykes)

f3c042e [Fri, 31 Jul 2015 19:23] updated gulpfile to not concatenate files in sub dirs (author: Alicia Sykes)

5d7a2d2 [Sat, 25 Jul 2015 14:40] Developed initial Gulpfile (author: Alicia Sykes)

(END)

3.2.3 git logs for 'find-region-from-position' module

8a981c1 [Wed, 3 Feb 2016 11:07] Wrote some documentation (author: Alicia Sykes)
36b0f54 [Wed, 3 Feb 2016 10:01] Finds and returns closest region, done and working (author: Alicia Sykes)
df79c8b [Wed, 3 Feb 2016 10:00] Wrote some unit tests (author: Alicia Sykes)
f9d604d [Tue, 2 Feb 2016 13:24] Finds a list of closest single directions lat or longs (author: Alicia Sykes)
81633d3 [Tue, 2 Feb 2016 12:46] Wrote convertCsvToJson function (author: Alicia Sykes)
fcaeea1 [Tue, 2 Feb 2016 12:46] Minify JS (author: Alicia Sykes)
0701398 [Mon, 1 Feb 2016 20:34] Set up blank module (author: Alicia Sykes)
7a8af49 [Mon, 1 Feb 2016 20:33] Added initial regions lat lng dataset (author: Alicia Sykes)
7e06f93 [Mon, 1 Feb 2016 20:33] Initial project setup NPM init (author: Alicia Sykes)
bb19c08 [Mon, 1 Feb 2016 20:32] Configured Gulp build environment (author: Alicia Sykes)
f49d0c9 [Mon, 1 Feb 2016 20:32] Added MIT License (author: Alicia Sykes)
41f3803 [Mon, 1 Feb 2016 20:31] Configured test environment, with Mocha, Travis, Chai and Istanbul (author: Alicia Sykes)
2aa273b [Mon, 1 Feb 2016 20:28] Project setup, gitignore and blank readme (author: Alicia Sykes)

3.2.4 Further logs

Further git logs are available for the other 8 modules developed and published as part of this project, however it would not have been feasible to include them all in this document (since there are over 1000 more commits!).

To generate additional logs, navigate into the appropriate directory, and run the command which is at the top of this document. Alternatively view the package on GitHub.

All git commits correspond to the time frame outlined in the methodology section.

4 APPENDIX FOUR

TEST RESULT EXAMPLES

This appendix document contains an example of a set of tests, and the results that prove this project followed the test strategy set out in the methodology.

Twitter Sentiment Visualisations

The research and development of a sentiment analysis module, and the implementation of it on real-time social media data, to generate a series of live visual representations of sentiment towards a specific topic or by location in order to find trends.

Alicia Sykes 12011471

Oxford Brookes University

4.1 CONTENTS

Introduction	151
Example Unit Tests	152
test/utils.test.coffee	152
test/main.test.coffee	154
Test Configuration	155
test/mocha.opts	155
.travis.yml	155
Running the tests	155
Exporting appropriate methods.....	155
Test Results	156
Example console output printed after tests are run.....	156
Example of the visual report generated for unit test results	157
Example of the visual report generated for coverage results	158
Example of the report generated for automated continuous integration testing.....	158
Example of summary of build status	159
Example of automated code review reports.....	159
Example of automated dependency checking report.....	160

4.2 INTRODUCTION

The purpose of this document is to show evidence that the final solution meets the following user stories relating to the test environment.

TSV-U085 – As a test environment I should run unit tests

TSV-U086 – As a test environment I should have integration tests

TSV-U087 – As a test environment I should be able to stub data as to not make external data requests

TSV-U088 – As a test environment I should use assertions for the unit tests

TSV-U089 – As a test environment I should show coverage test results

TSV-U090 – As a test environment I should check dependencies are up-to-date

TSV-U091 – As a test environment I should check code quality with automated code reviews

TSV-U092 – As a test environment I should do headless browser testing and HTTP service testing

4.3 EXAMPLE UNIT TESTS

The following unit tests are taken from the *sentiment-analysis* module

4.3.1 test/utlis.test.coffee

```
expect = require('chai').expect
process.env.NODE_ENV = 'test'

sentimentAnalysis = require('../index')._private

describe 'doesWordExist will return boolean weather word exists', ()->
  doesWordExist = sentimentAnalysis.doesWordExist

  it 'should return a boolean value', ()->
    expect(doesWordExist('coffee')).to.be.a('boolean')
    expect(doesWordExist('mocha')).to.be.a('boolean')
    expect(doesWordExist('java')).to.be.a('boolean')

  it 'should return true for words that exist', () ->
    expect(doesWordExist('woo')).to.be.true
    expect(doesWordExist('alive')).to.be.true
    expect(doesWordExist('awesome')).to.be.true
    expect(doesWordExist('anger')).to.be.true
    expect(doesWordExist('bright')).to.be.true
    expect(doesWordExist('love')).to.be.true
    expect(doesWordExist('easy')).to.be.true
    expect(doesWordExist('drunk')).to.be.true
    expect(doesWordExist('dumb')).to.be.true
    expect(doesWordExist('hacked')).to.be.true
    expect(doesWordExist('important')).to.be.true
    expect(doesWordExist('hug')).to.be.true
    expect(doesWordExist('itchy')).to.be.true
    expect(doesWordExist('laugh')).to.be.true
    expect(doesWordExist('stupid')).to.be.true
    expect(doesWordExist('bomb')).to.be.true

  it 'should return false for words that do not exist', () ->
    expect(doesWordExist('hello')).to.be.false
    expect(doesWordExist('world')).to.be.false
    expect(doesWordExist('everything')).to.be.false
    expect(doesWordExist('is')).to.be.false
    expect(doesWordExist('stupidness')).to.be.false
    expect(doesWordExist('acid')).to.be.false
    expect(doesWordExist('dinosaurs')).to.be.false
    expect(doesWordExist('laptop')).to.be.false
    expect(doesWordExist('pepsi')).to.be.false
    expect(doesWordExist('lorem')).to.be.false
    expect(doesWordExist('ipsum')).to.be.false
    expect(doesWordExist('squashed')).to.be.false
    expect(doesWordExist('watson')).to.be.false
    expect(doesWordExist('brain')).to.be.false

  it 'should not throw an error with funny values', ()->
    expect(doesWordExist(1)).to.be.a('boolean')
    expect(doesWordExist([])).to.be.a('boolean')
    expect(doesWordExist(true)).to.be.a('boolean')
    expect(doesWordExist(undefined)).to.be.a('boolean')
    expect(doesWordExist(1)).to.be.false
    expect(doesWordExist([])).to.be.false
    expect(doesWordExist(undefined)).to.be.false

describe 'getScoreOfWord method return a sentiment score for that word', ()->
  getScoreOfWord = sentimentAnalysis.getScoreOfWord
```

```

it 'should return an integer', ()->
  expect(getScoreOfWord('amazing')).to.be.a('number')
  expect(getScoreOfWord('warm')).to.be.a('number')
  expect(getScoreOfWord('yummy')).to.be.a('number')

it 'should be in a range of -5 to + 5', () ->
  expect(getScoreOfWord('nice')).to.be.above(-5).to.be.below(5)
  expect(getScoreOfWord('good')).to.be.below(5).to.be.below(5)
  expect(getScoreOfWord('great')).to.be.above(-5).to.be.below(5)
  expect(getScoreOfWord('awesome')).to.be.above(-5).to.be.below(5)

it 'should return 0 if word doesn\'t exist, rather than crashing', ()->
  expect(getScoreOfWord('batman')).equal(0)
  expect(getScoreOfWord('superman')).equal(0)
  expect(getScoreOfWord('spiderman')).equal(0)
  expect(getScoreOfWord('pepperpig')).equal(0)

it 'should return 0 if passed multiple words at a time that don\'t exist', ()->
  expect(getScoreOfWord('type error')).equal(0)
  expect(getScoreOfWord('everything is stupid')).equal(0)
  expect(getScoreOfWord('dinosaurs are awesome')).equal(0)

it 'should return actual positive score for positive words that exist', ()->
  expect(getScoreOfWord('united')).equal(1)
  expect(getScoreOfWord('unstoppable')).equal(2)
  expect(getScoreOfWord('excited')).equal(3)
  expect(getScoreOfWord('win')).equal(4)
  expect(getScoreOfWord('outstanding')).equal(5)

it 'should return actual negative score for negative words that exist', ()->
  expect(getScoreOfWord('fight')).equal(-1)
  expect(getScoreOfWord('fails')).equal(-2)
  expect(getScoreOfWord('evil')).equal(-3)
  expect(getScoreOfWord('fraud')).equal(-4)
  expect(getScoreOfWord('twat')).equal(-5)

it 'should return 0 for neutral words that exist', ()->
  expect(getScoreOfWord('some kind')).equal(0)
  # There is only 1 neutral result in the AFINN word list!

describe 'getWordsInSentence will transform a sentence into a clean array', ()->
  getWordsInSentence = sentimentAnalysis.getWordsInSentence

  it 'Should correctly turn a sentence into an array', ()->
    expect(getWordsInSentence('hello world')).eql(['hello', 'world'])
    expect(getWordsInSentence('this is a longer sentence'))
      .eql(['this', 'is', 'a', 'longer', 'sentence'])

  it 'Should normalise case', ()->
    expect(getWordsInSentence('HeLlO wOrLd')).eql(['hello', 'world'])
    expect(getWordsInSentence('JAVASCRIPT')).eql(['javascript'])

  it 'Should remove duplicates', ()->
    expect(getWordsInSentence('foo foo bar foo'))
      .eql(['foo', 'bar'])
    expect(getWordsInSentence('foo foo BAR Foo bAr foO bar foo'))
      .eql(['foo', 'bar', ])

  it 'Should remove blanks', ()->
    expect(getWordsInSentence('space      blank      '))
      .eql(['space', 'blank'])

  it 'Should remove special characters', ()->
    expect(getWordsInSentence('foo ! ^&*^&^%&^^&%^bar$$$^'))
      .eql(['foo', 'bar'])

describe 'removeDuplicates should remove duplicates from an array', () ->
  removeDuplicates = sentimentAnalysis.removeDuplicates

```

```

it 'should remove duplicates', () ->
  expect(removeDuplicates(['hello', 'world', 'hello', 'hello']))
    .eql(['hello', 'world'])

describe 'scaleScore should ensure the score is within the valid range', () ->
  scaleScore = sentimentAnalysis.scaleScore

  it 'should not be below -1', () ->
    expect(scaleScore(-1.2)).to.be.above(-1.01)
    expect(scaleScore(-38.8)).to.be.above(-1.01)
    expect(scaleScore(1.2)).to.be.above(-1.01)

  it 'should not be above +1', () ->
    expect(scaleScore(4.5)).to.be.below(1.01)
    expect(scaleScore(42)).to.be.below(1.01)
    expect(scaleScore(-1.2)).to.be.below(1.01)

  it 'should have 1 or 2 decimal places', () ->
    expect(scaleScore(1)).to.be.within(-1,+1);
    expect(scaleScore(-1)).to.be.within(-1,+1);
    expect(scaleScore(0)).to.be.within(-1,+1);
    expect(scaleScore(10)).to.be.within(-1,+1);
    expect(scaleScore(-1)).to.be.within(-1,+1);
    expect(scaleScore(-1.01)).to.be.within(-1,+1);
    expect(scaleScore(+1.0001)).to.be.within(-1,+1);
    expect(scaleScore(999999)).to.be.within(-1,+1);
    expect(scaleScore(-999999)).to.be.within(-1,+1);
    expect(scaleScore(-0)).to.be.within(-1,+1);
    expect(scaleScore(+0)).to.be.within(-1,+1);
    expect(scaleScore(42)).to.be.within(-1,+1);
    expect(scaleScore(3.1415926535897932)).to.be.within(-1,+1);
    expect(scaleScore(-273.15)).to.be.within(-1,+1);

```

4.3.2 test/main.test.coffee

```

expect = require('chai').expect
process.env.NODE_ENV = 'test'
sentimentAnalysis = require('../index').main

describe 'Check the modules basic functionality', ()->
  it 'should return an integer', () ->
    expect(sentimentAnalysis('lorem ipsum dolor seit amet'))
      .to.be.a('number')
    expect(sentimentAnalysis('foo bar')).to.not.be.undefined;

  it 'Should return the correct sentiment value for negative sentences', () ->
    expect(sentimentAnalysis('I hate everything, everything is stupid')).equal(-0.5)
    expect(sentimentAnalysis('London is gloomy today because of all the smog')).equal(-0.4)
    expect(sentimentAnalysis('He was captured and put into slavery')).equal(-0.3)
    expect(sentimentAnalysis('Windows is very unstable')).equal(-0.2)
    expect(sentimentAnalysis('The slug was tired, he felt sluggish')).equal(-0.2)

  it 'Should return the correct sentiment value for positive sentences', () ->
    expect(sentimentAnalysis('Today is a wonderful amazing awesome day')).equal(1)
    expect(sentimentAnalysis('I am so grateful for all the presents, thank you!')).equal(0.5)

  it 'Should not return a score greater than 1 or smaller than -1', () ->
    expect(sentimentAnalysis('happy happy amazing awesome cool'))
      .to.be.above(-1.1).to.be.below(1.1)
    expect(sentimentAnalysis('crap crap crap crap'))
      .to.be.above(-1.1).to.be.below(1.1)

  it 'Should be able to cope with weird inputs and never crash', ()->

```

4.4 TEST CONFIGURATION

4.4.1 test/mocha.opts

```
--compilers coffee:coffee-script/register
--reporter spec
```

4.4.2 .travis.yml

```
language: node_js
node_js:
  - "0.12"
  - "0.10"

before_script:
  - "npm i -g mocha"
```

4.4.3 Running the tests

All tests can be run by running the command 'npm test' or 'gulp test'

The gulp task which runs all tests is as follows:

```
/* Run unit tests and generate coverage report */
gulp.task('test', function (cb) {
  gulp.src(['./index.js'])
    .pipe(istanbul())
    .pipe(istanbul.hookRequire())
    .on('finish', function () {
      gulp.src('./test/**/*.coffee', {read: false})
        .pipe(mocha({ reporter: 'spec' }))
        .pipe(istanbul.writeReports({reporters: ['text-summary', 'lcov']}))
        // .pipe(istanbul.enforceThresholds({ thresholds: { global: 90 } }))
        .on('end', cb);
    });
});
```

4.4.4 Exporting appropriate methods

If developing in the test environment (as opposed to production), we also export private methods, which allows them to be unit tested. For example:

```
if process.env.NODE_ENV == 'test'
  module.exports =
    main: analyseSentence
    _private:
      scaleScore: scaleScore
      doesWordExist: doesWordExist
      getScoreOfWord: getScoreOfWord
      removeDuplicates: removeDuplicates
      getWordsInSentence: getWordsInSentence
```

4.5 TEST RESULTS

4.5.1 Example console output printed after tests are run

```
C:\Users\Alicia\Dropbox\Coding\Nodejs\sentiment-analysis (master)
λ npm test

> sentiment-analysis@0.1.1 test C:\Users\Alicia\Dropbox\Coding\Nodejs\sentiment-analysis
> gulp test

[13:08:03] Using gulpfile ~\Dropbox\Coding\Nodejs\sentiment-analysis\gulpfile.js
[13:08:03] Starting 'test'...

Check the modules basic functionality
  ✓ should return an integer
  ✓ Should return the correct sentiment value for negative sentences
  ✓ Should return the correct sentiment value for positive sentences
  ✓ Should not return a score greater than 1 or smaller than -1
  ✓ Should be able to cope with weird inputs and never crash

doesWordExist will return boolean whether word exists
  ✓ should return a boolean value
  ✓ should return true for words that exist
  ✓ should return false for words that do not exist
  ✓ should not throw an error with funny values

getScoreOfWord method return a sentiment score for that word
  ✓ should return an integer
  ✓ should be in a range of -5 to + 5
  ✓ should return 0 if word doesn't exist, rather than crashing
  ✓ should return 0 if passed multiple words at a time that don't exist
  ✓ should return actual positive score for positive words that exist
  ✓ should return actual negative score for negative words that exist
  ✓ should return 0 for neutral words that exist

getWordsInSentence will transform a sentence into a clean array
  ✓ Should correctly turn a sentence into an array
  ✓ Should normalise case
  ✓ Should remove duplicates
  ✓ Should remove blanks
  ✓ Should remove special characters

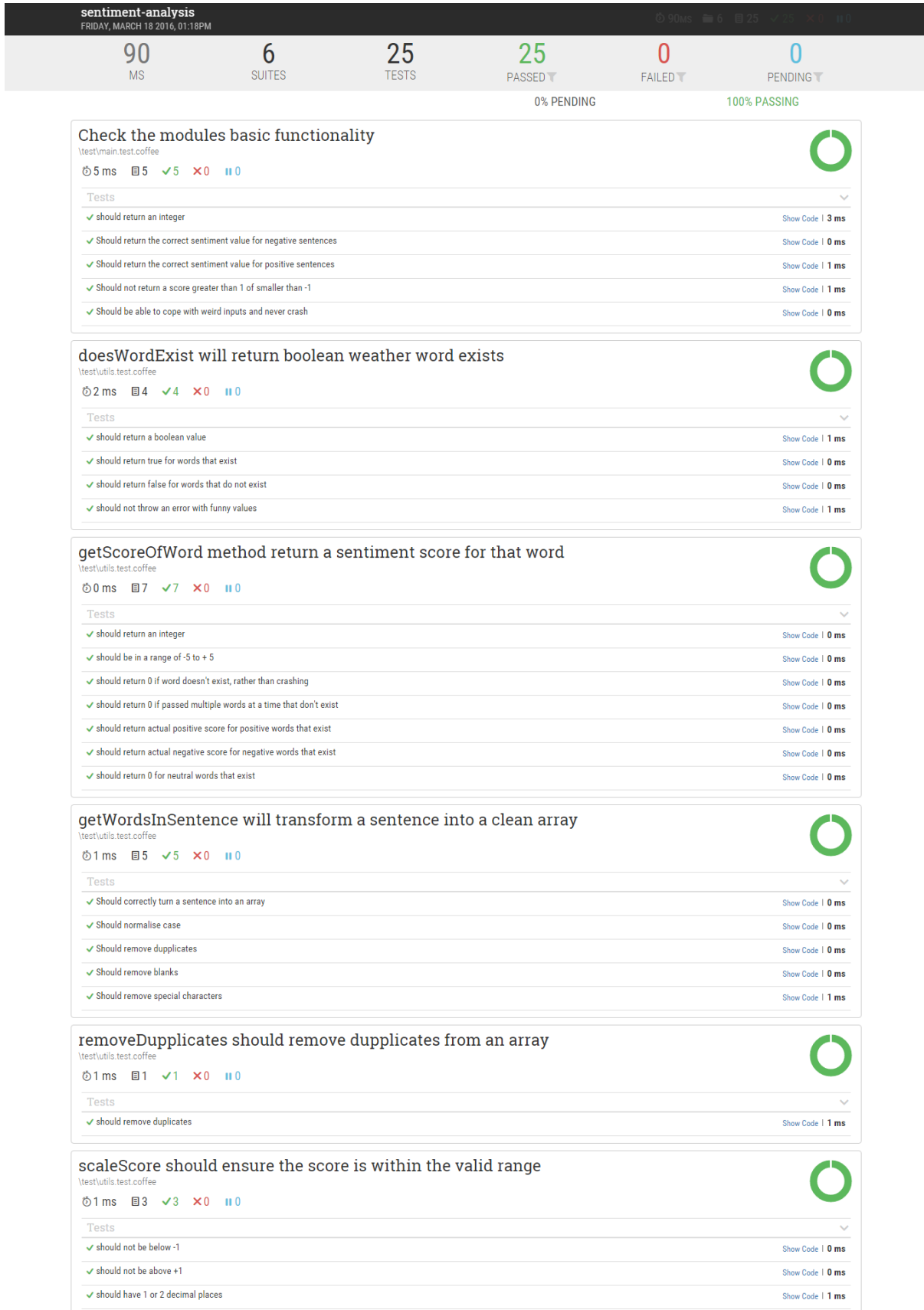
removeDuplicates should remove duplicates from an array
  ✓ should remove duplicates

scaleScore should ensure the score is within the valid range
  ✓ should not be below -1
  ✓ should not be above +1
  ✓ should have 1 or 2 decimal places

25 passing (97ms)

===== Coverage summary =====
Statements   : 97.96% ( 48/49 )
Branches     : 72.73% ( 16/22 )
Functions    : 100% ( 8/8 )
Lines       : 97.96% ( 48/49 )
=====
[13:08:04] Finished 'test' after 1.45 s
```


4.5.2 Example of the visual report generated for unit test results



4.5.3 Example of the visual report generated for coverage results

Code coverage report for **sentiment-analysis**

Statements: **87.5%** (14 / 16) Branches: **75%** (3 / 4) Functions: **60%** (3 / 5) Lines: **87.5%** (14 / 16) Ignored: none

[All files](#) » sentiment-analysis/

File	Statements	Branches	Functions	Lines
index.js	87.5% (14 / 16)	75% (3 / 4)	60% (3 / 5)	87.5% (14 / 16)

Generated by [istanbul](#) at Wed Sep 30 2015 12:39:33 GMT+0100 (GMT Daylight Time)

4.5.4 Example of the report generated for automated continuous integration testing

Travis CI [Blog](#) [Status](#) [Help](#) Alicia Sykes

Search all repositories

Lissy93 / sentiment-analysis build passing

Current Branches Build History Pull Requests Requests More options



✓	2cc7919	5 months ago	Bumped to V0.1.1	# 29	Build created successfully
✓	722ea34	5 months ago	Wrote unit tests for scaleScore, 100% coverage #res	# 28	Build created successfully
✓	d00f13c	5 months ago	Updated the documentation for new sentiment ran	# 27	Build created successfully
✓	-	6 months ago	-	-	missing commit
✓	d60e3a0	6 months ago	V0.1.0	# 26	Build created successfully
✓	72fe1e	6 months ago	Added continuous integration testing config file for	# 25	Build created successfully
✓	82d9a58	6 months ago	V0.1.0	# 24	Build created successfully
✓	9745ee6	6 months ago	V0.1.0	# 23	Build created successfully
✓	f11772e	6 months ago	Merge pull request #1 from Lissy93/master	# 22	Build created successfully
✓	af76adc	6 months ago	reversed pull request	# 21	Build created successfully
✓	f11772e	6 months ago	Merge pull request #1 from Lissy93/master	# 20	Build created successfully
✓	82d9a58	6 months ago	V0.1.0	# 19	Build created successfully
✓	#1 71f4b57	6 months ago	V0.1.0	# 18	Build created successfully
✓	#1 1763a39	6 months ago	Merged with master	# 17	Build created successfully
✓	2584d44	6 months ago	Merged with master	# 16	Build created successfully
✓	9745ee6	6 months ago	V0.1.0	# 15	Build created successfully
✓	0045211	6 months ago	Wrote usage documentation	# 14	Build created successfully
✓	6b7a29e	6 months ago	Changed path of words list to __dirname	# 13	Build created successfully
✓	1893a21	6 months ago	Finished main class should pass all tests	# 12	Build created successfully
✓	974c982	6 months ago	Included test cases for the overall main function	# 11	Build created successfully
✓	824893c	6 months ago	Test removeDuplicates method	# 10	Build created successfully
✓	68d2e56	6 months ago	Wrote getWordsInSentence and checked it passes !	# 9	Build created successfully
✓	b5cf9e7	6 months ago	Wrote unit tests for getWordsInSentence method	# 8	Build created successfully
✓	9c66b68	6 months ago	debug statement	# 7	Build created successfully
✓	a95056f	6 months ago	Wrote unit tests for getScoreOfWord	# 6	Build created successfully


4.5.5 Example of summary of build status


Lissy93 / twitter-sentiment-visualisation  build passing


[Current](#) [Branches](#) [Build History](#) [Pull Requests](#) More options 


✓ **dev** New thumbnail for World Now page


-  Commit 0a9c124
-  Compare c65a4b2..0a9c124

 Alicia Sykes authored and committed







➔ #122 passed 

 Elapsed time 3 min 31 sec

 Total time 5 min 47 sec

 2 days ago

Build Jobs

✓	# 122.1	 </> Node.js: 0.12	 no environment variables set	 3 min 31 sec
✓	# 122.2	 </> Node.js: 0.10	 no environment variables set	 2 min 16 sec


4.5.6 Example of automated code review reports

Complex method awesomeizeScripts (complexity = 25)

```
30 function awesomeizeScripts(srcPath, resPath){
31
32     /* Filters to be applied (so that different operations can be done on different files) */
33     var bundleFilter = filter(['*', '!**/*-main.{js,coffee}', '!**/*-main.{js,coffee}']);
34     var coffeeFilter = filter('**/*.coffee', {restore: true}); // MUST be declared here in c
```

[View more](#)

Found in [tasks/scripts.js](#)

'mobileRoute' is not defined. (Line 71) 71 mobileRoute.get('/', function(req, res, next) { 


Found in [app.js](#)

'next' is defined but never used. (Line 72) 72 return res.render('error', { 

Found in [app.js](#)

'next' is defined but never used. (Line 100) 100 res.status(err.status || 500); 

Found in [app.js](#)

Missing semicolon. (Line 3) 3 "*/\r\n\n*/ MIT License. Read full license at: https://\goo. 

Found in [tasks/config.js](#)

4.5.7 Example of automated dependency checking report

LISSY93 - TWITTER-SENTIMENT-VISUALISATION 0.0.1 dependencies up to date

A series of data visualisations showing overall sentiment from Tweets by location and/or topic

DEPENDENCIES
DEVDEPENDENCIES
LIST TREE

■ 22 Dependencies total
 ■ 22 Up to date
■ 0 Pinned, out of date
■ 0 Out of date

DEPENDENCY	REQUIRED	STABLE	LATEST	STATUS
body-parser	^1.14.0	1.15.0	1.15.0	■
coffee-script	^1.9.3	1.10.0	1.10.0	■
cookie-parser	^1.4.0	1.4.1	1.4.1	■
debug	^2.2.0	2.2.0	2.2.0	■
express	^4.13.3	4.13.4	5.0.0-alpha.2	■
fetch-tweets	^0.1.7	0.1.7	0.1.7	■
find-region-from-location	git+https://github.com/Lissy93/find-region-from-location.git			■
haven-entity-extraction	git://github.com/Lissy93/haven-entity-extraction.git			■
haven-sentiment-analysis	git://github.com/Lissy93/haven-sentiment-analysis.git			■
jade	^1.11.0	1.11.0	1.11.0	■
mobile-redirect	0.0.1	0.0.1	0.0.1	■
moment	^2.11.2	2.12.0	2.12.0	■
mongoose	^4.1.6	4.4.8	4.4.8	■
morgan	^1.6.1	1.7.0	1.7.0	■
place-lookup	0.0.2	0.0.2	0.0.2	■
q	^1.4.1	1.4.1	2.0.3	■
remove-words	^0.2.0	0.2.0	0.2.0	■
sentiment-analysis	^0.1.1	0.1.1	0.1.1	■
serve-favicon	^2.3.0	2.3.0	2.3.0	■
socket.io	^1.3.6	1.4.5	1.4.5	■
stream-tweets	^1.1.0	1.1.0	1.1.0	■
watson-developer-cloud	^1.2.3	1.3.0	1.3.0	■

■ 22 Dependencies total
 ■ 22 Up to date
■ 0 Pinned, out of date
■ 0 Out of date

5 APPENDIX FIVE - USER EXPERIENCE SURVEY

Which of the following best describe your use for sentiment sweep?

- a) Reviewing marketing campaign effectiveness
- b) Predicting upcoming changes in stock prices for investment
- c) Comparing consumer opinions on brands and products
- d) Predicting future political leaders
- e) Checking and analysing major disruptions or disputes
- f) Looking at general geographical positive: negative ratios
- g) Following the success/failure of a sports team or person in real-time through an event
- h) Other: _____

Age Range: >18 18 – 25 25 – 65 65+

Technical Ability (0 = know nothing, 5 = expert): 0 1 2 3 4 5

Search Page: Overall Sentiment

Is it clear what overall result is conveyed?	1	2	3	4	5	-
How useful is it for you to see these results?	1	2	3	4	5	-
How accurate do you think these results are?	1	2	3	4	5	-
Is the data visualization/ chart easy to interpret?	1	2	3	4	5	-

Comments:

Tone Identification

Is it clear what overall result is conveyed?	1	2	3	4	5	-
How useful is it for you to see these results?	1	2	3	4	5	-
How accurate do you think these results are?	1	2	3	4	5	-
Is the data visualization/ chart easy to interpret?	1	2	3	4	5	-

Comments:

Entity Extraction

Is it clear what overall result is conveyed?	1	2	3	4	5	-
How useful is it for you to see these results?	1	2	3	4	5	-
How accurate do you think these results are?	1	2	3	4	5	-
Is the data visualization/ chart easy to interpret?	1	2	3	4	5	-

Comments:

Topic Comparison

Is it clear what overall result is conveyed?	1	2	3	4	5	-
How useful is it for you to see these results?	1	2	3	4	5	-
How accurate do you think these results are?	1	2	3	4	5	-
Is the data visualization/ chart easy to interpret?	1	2	3	4	5	-

Comments:

Raw Tweets

How useful is it for you to see Tweets in this form?	1	2	3	4	5	-
Is the layout and use of the page easy to understand?	1	2	3	4	5	-

Comments:

Heat Map

Is it clear the heat map clear to understand?	1	2	3	4	5	-
Can you interpret which areas are more/less positive?	1	2	3	4	5	-

Comments:

Regional Map

Is it clear to see which regions are more/ less positive? 1 2 3 4 5 -

Comments:

3D Globe

Overall rating: 1 2 3 4 5 -

Comments:

Word Cloud and Scatter Plot

Is it clear what overall result is conveyed? 1 2 3 4 5 -

How useful is it for you to see these results? 1 2 3 4 5 -

How accurate do you think these results are? 1 2 3 4 5 -

Is the data visualization/ chart easy to interpret? 1 2 3 4 5 -

Comments:

Trending

Overall Rating: 1 2 3 4 5 -

Comments:

Timeline

Overall Rating: 1 2 3 4 5 -

Comments:

Home page

What does the application do? (how well did they grasp the concept?)

0 1 2 3 4 5 -

How clear are the two entry routes (search or scroll) on the homepage?

0 1 2 3 4 5 -

What do you think of the home background? (select all that apply)

- It shows an interesting snapshot of the overall sentiment on twitter right now
- It's cool, but kind of useless
- I have no idea what it is

- I like how it is updating in real-time, showing live tweets
- I hate how it keeps changing, it's so distracting!
- I have no opinion on it's real time coolness

- The colours are terrible
- It's nice and colourful
- I have no opinion on the colours

Any Final Comments?

6 APPENDIX SIX

CODE LISTING

Twitter Sentiment Visualisations

The research and development of a sentiment analysis module, and the implementation of it on real-time social media data, to generate a series of live visual representations of sentiment towards a specific topic or by location in order to find trends.

Alicia Sykes 12011471
Oxford Brookes University

6.1 DIRECTORY STRUCTURE OF THE SOURCE APPLICATION

client-side-sauce

/graphics

/scripts

/styles

server-side-sauce

/api-routes

/config

/models

/routes

/utils

views

/components

/pages

/sections

dev-tasks

6.2 DIRECTORY STRUCTURE OF THE PRODUCTION APPLICATION

config

/app-config.js

/keys.js

models

/Tweet.js

public

/data

/images

/javascripts

/stylesheets

/favicon.ico

routes

utils

views

The following sections contains the code from just the source application.

6.3 TABLE OF CONTENTS

Application root, main and config files	172
./app.js (main entry point to application)	172
./.gitignore	174
./.travis.yml	174
./gulpfile.js	174
./package.json	175
./bower.json	177
Server Side scripts – utility code	178
utils/async-tweets.coffee	178
utils/fetch-sentiment-tweets.coffee	178
utils/make-complete-tweets.coffee	179
utils/stream-handler.coffee	180
utils/format-for-keyword-vis.coffee	181
utils/format-tweets-for-map.coffee	182
utils/make-click-words.coffee	184
utils/Watson-tone-analyser.coffee	184
utils/make-summary-sentence.coffee	185
utils/make-timeline-data.coffee	186
Models	188
models/Tweet.coffee	188
api-routes	189
api-routes/db-api.coffee	189
api-routes/antity-api.coffee	189
api-routes/tone-api.coffee	190
api-routes/trending-api.coffee	191
Routes	192
routes/about.coffee	192
routes/index.coffee	192
routes/now.coffee	192
routes/real-time-dash.coffee	192
routes/break-down.coffee	193
routes/timeline.coffee	194

routes/region-map.coffee	195
routes/tone-analyzer.coffee.....	196
routes/trending.coffee.....	196
routes/sa-comparison.coffee	197
routes/search.coffee.....	198
routes/word-cloud.coffee	199
routes/text-tweets.coffee	200
routes/word-plot.coffee	201
config	202
config/app-config.coffee	202
config/api-keys.coffee.....	202
Client Side Scripts	203
map/map-main.coffee	203
map/heatmap-module.coffee	204
map/live-interactions-module.coffee	206
Map/marker-cluster-module.coffee	207
map/options-module.coffee	209
map/socket-module.coffee	209
map/autocomplete-module.coffee.....	209
map/search-module.coffee	210
map/theme-module.coffee	210
globe/globe-main.coffee.....	212
globe/socket-module.coffee	212
globe/configure-globe.coffee	212
now/now-main.coffee	214
now/now-count-module.coffee	215
now/now-bars-module.coffee.....	216
now/now-map-modue.coffee	217
now/now-top-tweets-module.coffee	218
visualisations/break-down.coffee	219
visualisations/comparer.coffee	220
visualisations/comparison.coffee	220
visualisations/comparison-for-search.coffee	223
visualisations/entity-extraction.coffee	223
visualisations/entity-summary.coffee.....	225
visualisations/gauge-module.coffee	225

visualisations/hexagons-module.coffee.....	226
visualisations/homepage.coffee.....	228
visualisations/radar-module.coffee.....	229
visualisations/real-time-dash.coffee.....	230
visualisations/regions-main.coffee.....	231
visualisations/render-tone-bars.coffee.....	232
visualisations/search-summary-main.coffee.....	232
visualisations/scatter-words-main.coffee.....	233
visualisations/earch-summary-main.coffee.....	236
visualisations/text-tweets.coffee.....	236
visualisations/timeline-main.coffee.....	237
visualisations/tone-analysis-main.coffee.....	238
visualisations/trending.coffee.....	239
visualisations/word-cloud-main.coffee.....	241
/loading.coffee.....	242
/page-controls-module.coffee.....	242
Styles (client-side).....	243
styles/constants.less.....	243
styles/general.less.....	243
styles/loading.css.....	244
styles/charts.less.....	245
styles/components.less.....	249
styles/map.less.....	252
styles/materialize.less.....	253
Views (run on server-side, rendered on client-side).....	254
views/layout.jade.....	254
views/error.jade.....	254
views/component-divider.jade.....	254
views/component-spinner.jade.....	255
views/footer.jade.....	255
views/footer-scripts.jade.....	256
views/head.jade.....	256
views/side-nav.jade.....	256
views/navbar.jade.....	256
index.jade.....	257
views/page-about.jade.....	259

views/page-breakdown.jade	262
views/page-cloud.jade	263
views/page-comparer.jade	264
views/page-sa-comparison.jade	266
views/page-entity-extraction.jade.....	267
views/page-word-plot.jade	268
views/page-globe.jade	269
views/page-hexagons.jade.....	269
views/page-realtime.jade	270
views/page-realtime.jade	270
views/page-trending.jade	271
Development and Build Files	272
./gulpfile.js	272
tasks/browser-sync.js	272
tasks/browserify.js	273
tasks/generate-config.js.....	273
tasks/scripts.js	274
tasks/styles.js	275
tasks/test.js	276
tasks/watch.js	276
tasks/nodemon.js	277
tasks/quick-coffeescript.js.....	277
tasks/images.js	277
tasks/clean.js.....	278
tasks/build.js	278
tasks/config.js	278
Module 1 – Sentiment Analysis	279
index.coffee.....	279
.gitignore	280
.travis.yml	280
dev.js	280
index.js (compiled).....	280
package.json.....	281
gulpfile.js.....	282
readme.md.....	283
test/mocha.opts	284

test/utlis.test.coffee	284
test/main.test.coffee	286
module 2- fetch tweets	287
module 3 – stream tweets.....	289
module 4 – remove words.....	291
module 5 - place lookup	292
module 6 – tweet location.....	293
module 7 – hp haven SA.....	293
module 8 – HP Haven Extract Entities.....	294
module 9 – find region from location	295

6.4 APPLICATION ROOT, MAIN AND CONFIG FILES

6.4.1 ./app.js (main entry point to application)

```
/* Include necessary node modules */
var express      = require('express');
var path         = require('path');
var favicon      = require('serve-favicon');
var logger       = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser   = require('body-parser');
var mongoose     = require('mongoose');
var http         = require('http');
var mdirect      = require('mobile-redirect');
var streamTweets = require('stream-tweets');
var config       = require('./config/app-config');
var streamHandler = require('./utils/stream-handler');

/* Create Express server and configure socket.io */
var app = express();
var server = http.createServer(app);
var io = require('socket.io').listen(server);
server.listen(config.server.port, function(){
  console.log('Express server listening on port ' + config.server.port);
});

/* view engine setup */
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');

/* Set up other Express bits and bobs */
app.use(favicon(path.join(__dirname, 'public', 'favicon.ico')));
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));
app.use('/bower_components', express.static(__dirname + '/bower_components'));
app.use(mdirect({redirectPath: '/mobile'}));

/* Connect to MongoDB */
mongoose.connect(config.db.URL);

/* Specify which route files to use */
app.use('/', require('./routes/index'));
app.use('/search', require('./routes/search'));
app.use('/map', require('./routes/map'));
app.use('/region-map', require('./routes/region-map'));
app.use('/globe', require('./routes/globe'));
app.use('/timeline', require('./routes/timeline'));
app.use('/about', require('./routes/about'));
app.use('/text-tweets', require('./routes/text-tweets'));
app.use('/break-down', require('./routes/break-down'));
app.use('/hexagons', require('./routes/hexagons'));
app.use('/comparer', require('./routes/comparer'));
app.use('/word-cloud', require('./routes/word-cloud'));
app.use('/trending', require('./routes/trending'));
app.use('/tone-analyzer', require('./routes/tone-analyzer'));
app.use('/sa-comparison', require('./routes/sa-comparison'));
app.use('/word-scatter-plot', require('./routes/word-plot'));
app.use('/entity-extraction', require('./routes/entity-extraction'));
//app.use('/real-time-dashboard', require('./routes/real-time'));
```



```

app.use('/now',      require('./routes/now'));
app.use('/api/tone', require('./routes/tone-api'));
app.use('/api/entity', require('./routes/entity-api'));
app.use('/api/db',   require('./routes/db-api'));
app.use('/api/trending', require('./routes/trending-api'));

mobileRoute = express.Router();
mobileRoute.get('/', function(req, res, next) {
  return res.render('error', {
    message: 'Not compatible with your device :\'(',
    error: {}
  });
});

app.use('/mobile', mobileRoute);

/* Set a stream listener for tweets matching tracking keywords */
var credentials = require('./config/keys').twitter;
var twit = new streamTweets(credentials);
var world = [-180,-90,180,90];
stream = twit.stream({ locations: world}, function(stream) {
  streamHandler(stream,io); });

/* catch 404 and forward to error handler */
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

/*-- error handlers -- */

/* development error handler (will print stacktrace) */
if (app.get('env') === 'development') {
  app.use(function(err, req, res, next) {
    res.status(err.status || 500);
    res.render('error', {
      message: err.message,
      error: err
    });
  });
}

/* production error handler (no stacktraces leaked to user) */
app.use(function(err, req, res) {
  res.status(err.status || 500);
  res.render('error', {
    message: err.message,
    error: {}
  });
});

module.exports = app;

```

6.4.2 `./.gitignore`

```
# Packages and libraries
node_modules
bower_components

# IDE files and logs
.idea
*log
npm-debug.log.*

# Test reports and results
reports
coverage

# API keys
**/keys.*

# Built files
models/*.js
utils/*.js
config/*.js
routes/*.js
public/javascripts
public/stylesheets
```

6.4.3 `./.travis.yml`

```
language: node_js
node_js:
  - "0.12"
  - "0.10"

before_script:
  - "npm i -g mocha"

notifications:
  slack:
    on_success: change
```

6.4.4 `./gulpfile.js`

```
/**
 * @author Alicia Sykes <alicia@aliciasykes.com> June 2015
 * To run script run "gulp" in the command line
 * My super amazing gulp setup, does EVERYTHING cool
 * possible to do to turns already awesome code into
 * even more awesomely efficient code
 *
 * The Gulp tasks are divided into the files in the './tasks/' directory
 * Read the build documentation at './docs/building.md'
 */

// Include gulp and require directory module
var gulp = require('gulp');
var reqdir = require('require-dir');

reqdir('./tasks'); // Include the folders containing ALL gulp tasks

gulp.task('default', ['start']); // Default task
```

6.4.5 ./package.json

```
{
  "name": "twitter-sentiment-visualisation",
  "description":
    "A series of data visualisations showing overall sentiment
    from Tweets by location and/or topic",
  "author": "Alicia Sykes <alicia@aliciasykes.com>",
  "version": "0.0.1",
  "private": true,

  "scripts": {
    "start": "node ./bin/www",
    "test": "gulp test",
    "build": "gulp build",
    "config": "gulp generate-config && start notepad 'config\\src\\keys.coffee'",
    "cover":
      "istanbul cover node_modules/mocha/bin/_mocha
      --dir ./reports/coverage-reports"
  },

  "main": "app.js",
  "repository": {
    "type": "git",
    "url": "https://github.com/Lissy93/twitter-sentiment-visualisation"
  },

  "bugs": {
    "url": "https://github.com/Lissy93/twitter-sentiment-visualisation/issues"
  },

  "license": "MIT",

  "dependencies": {
    "body-parser": "^1.14.0",
    "coffee-script": "^1.9.3",
    "cookie-parser": "^1.4.0",
    "debug": "^2.2.0",
    "express": "^4.13.3",
    "fetch-tweets": "^0.1.7",
    "find-region-from-location":
      "git+https://github.com/Lissy93/find-region-from-location.git",
    "haven-entity-extraction":
      "git://github.com/Lissy93/haven-entity-extraction.git",
    "haven-sentiment-analysis":
      "git://github.com/Lissy93/haven-sentiment-analysis.git",
    "jade": "^1.11.0",
    "mobile-redirect": "0.0.1",
    "moment": "^2.11.2",
    "mongoose": "^4.1.6",
    "morgan": "^1.6.1",
    "place-lookup": "0.0.2",
    "q": "^1.4.1",
    "remove-words": "^0.2.0",
    "sentiment-analysis": "^0.1.1",
    "serve-favicon": "^2.3.0",
    "socket.io": "^1.3.6",
    "stream-tweets": "^1.1.0",
    "watson-developer-cloud": "^1.2.3"
  },
}
```

```

"devDependencies": {
  "browser-sync": "^2.8.2",
  "browserify": ^13.0.0",
  "chai": ^3.2.0",
  "coffeeify": ^2.0.1",
  "debowerify": ^1.3.1",
  "del": ^2.2.0",
  "event-stream": ^3.3.1",
  "glob": ^7.0.0",
  "gulp": ^3.9.0",
  "gulp-changed": ^1.2.1",
  "gulp-coffee": ^2.3.1",
  "gulp-coffeelint": ^0.6.0",
  "gulp-concat": ^2.6.0",
  "gulp-csslint": ^0.2.0",
  "gulp-filesize": 0.0.6",
  "gulp-filter": ^3.0.0",
  "gulp-footer": ^1.0.5",
  "gulp-if": ^2.0.0",
  "gulp-istanbul": ^0.10.0",
  "gulp-jshint": ^2.0.0",
  "gulp-less": ^3.0.3",
  "gulp-minify-css": ^1.2.0",
  "gulp-mocha": ^2.1.3",
  "gulp-nodemon": ^2.0.3",
  "gulp-recess": ^1.1.2",
  "gulp-rename": ^1.2.2",
  "gulp-sourcemaps": ^1.5.2",
  "gulp-uglify": ^1.2.0",
  "gulp-uncss": ^1.0.2",
  "gulp-util": ^3.0.6",
  "istanbul": ^0.4.2",
  "jshint-stylish": ^2.0.1",
  "lodash": ^4.3.0",
  "mocha": ^2.2.5",
  "mochawesome": ^1.2.0",
  "phantom": ^1.0.0",
  "reactify": ^1.1.1",
  "require-dir": ^0.3.0",
  "run-sequence": ^1.1.3",
  "sinon": ^1.15.4",
  "supertest": ^1.0.1",
  "vinyl-buffer": ^1.0.0",
  "vinyl-source-stream": ^1.1.0",
  "yargs": ^3.25.0
}
}

```

6.4.6 ./bower.json

```
{
  "name": "twitter-sentiment-visualisation",
  "version": "0.0.0",
  "authors": [
    "alicia@aliciasykes.com"
  ],
  "description": "A series of data visualisations showing overall sentiment from
Tweets by location and/or topic",
  "main": "app.js",
  "moduleType": [
    "node"
  ],
  "keywords": [
    "awesome",
    "twitter",
    "sentiment-analysis",
    "d3",
    "visualisation",
    "data"
  ],
  "license": "MIT",
  "homepage": "http://twitter-sentiment-visualisation.as93.net",
  "ignore": [
    "**/*.*",
    "node_modules",
    "bower_components",
    "test",
    "tests"
  ],
  "dependencies": {
    "d3": "~3.5.16",
    "google-chart": "GoogleWebComponents/google-chart#~1.0.4",
    "google-map": "GoogleWebComponents/google-map#~1.0.3",
    "jquery": "~2.1.4",
    "markerclusterer_compiled": "https://raw.githubusercontent.com/googlemaps/js-
marker-clusterer/gh-pages/src/markerclusterer_compiled.js",
    "materialize": "~0.97.5",
    "Detector": "https://raw.githubusercontent.com/dataarts/webgl-
globe/master/globe/third-party/Detector.js",
    "three.min": "https://raw.githubusercontent.com/dataarts/webgl-
globe/master/globe/third-party/three.min.js",
    "Tween": "https://raw.githubusercontent.com/dataarts/webgl-
globe/master/globe/third-party/Tween.js",
    "globe": "https://raw.githubusercontent.com/dataarts/webgl-
globe/master/globe/globe.js",
    "socket.io": "https://raw.githubusercontent.com/socketio/socket.io-
client/master/socket.io.js",
    "d3-word-cloud": "https://jardindesconnaissances.googlecode.com/svn-
history/r82/trunk/public/js/d3.layout.cloud.js",
    "tipsy":
"https://cdnjs.cloudflare.com/ajax/libs/jquery.tipsy/1.0.2/jquery.tipsy.js",
    "nvd3": "~1.8.2",
    "sankey.js": "https://raw.githubusercontent.com/d3/d3-
plugins/master/sankey/sankey.js",
    "hexbin": "http://d3js.org/d3.hexbin.v0.min.js",
    "d3tip": "https://raw.githubusercontent.com/Caged/d3-tip/master/index.js",
    "c3": "~0.4.10",
    "radar-chart-d3": "https://github.com/alangrafu/radar-chart-d3.git#~1.2.1",
    "Materialize": "materialize#~0.97.5",
    "jquery-autocomplete": "Autocomplete#~0.1.9",
    "bullet.js":
"https://gist.githubusercontent.com/mbostock/4061961/raw/6eb742223b9795260ba6215019
6ed0ae4a461e39/bullet.js"
  }
}
```

6.5 SERVER SIDE SCRIPTS – UTILITY CODE

6.5.1 utils/async-tweets.coffee

```
fetchSentimentTweets = require './utils/fetch-sentiment-tweets'  
q = require 'q'  
  
makeTweetPromiseArr = (searchTerm) ->  
  deferredTweetResults = q.defer()  
  fetchSentimentTweets searchTerm, (results) -> # Fetch all data for one Tweet  
    deferredTweetResults.resolve(results)  
  deferredTweetResults.promise  
  
makeRequests = (searchTerms, completeAction) ->  
  results = []  
  promises = []  
  for term in searchTerms then promises.push makeTweetPromiseArr term  
  q.all(promises).spread -> # When all the twitter promises have returned  
    for argument, index in arguments  
      results.push searchTerm: searchTerms[index], tweets: argument  
    completeAction results # Done!  
  
module.exports = makeRequests
```

6.5.2 utils/fetch-sentiment-tweets.coffee

```
# Include relevant node modules  
FetchTweets = require 'fetch-tweets'  
sentiment = require 'sentiment-analysis'  
removeWords = require 'remove-words'  
twitterKey = require('./config/keys').twitter  
Tweet = require './models/Tweet'  
  
module.exports = (searchTerm, callback) ->  
  
  if searchTerm == '' then Tweet.getAllTweets (results) ->  
    format results, callback  
  
  else (new FetchTweets twitterKey).byTopic searchTerm, (results) ->  
    format results, callback  
  
  format = (results, callback) ->  
  
    # Assign Sentiments  
    for tweet in results then tweet.sentiment = sentiment tweet.body  
  
    # Assign keywords  
    for tweet in results then tweet.keywords = removeWords tweet.body  
  
    # Find average sentiment  
    total = 0  
    for tweet in results then total += tweet.sentiment  
    averageSentiment = total / results.length  
    averageSentiment = Math.round(averageSentiment*100)/100  
  
    # Done, call callback with results and sentiment average  
    callback results, averageSentiment
```

6.5.3 utils/make-complete-tweets.coffee

```
# Include relevant node modules
FetchTweets = require 'fetch-tweets'
placeLookup = require 'place-lookup'
sentiment = require 'sentiment-analysis'
removeWords = require 'remove-words'
q = require 'q'

class GetGeoSentimentTweets

  constructor: (@twitterApiKeys, @placesApiKey) ->

# Function fetches Tweets from Twitter API, returning a deferred promise
fetchFromTwitter = (query, twitterApiKeys) ->
  fetchTweets = new FetchTweets(twitterApiKeys)
  deferredTweets = q.defer()
  fetchTweets.byTopic query, (results) ->
    deferredTweets.resolve(results)
  deferredTweets.promise

# Fetches place data (lat and long) from Google places api, return promise
findPlaceInfo = (locationObject, placesApiKey) ->
  deferredLocation = q.defer()
  if locationObject.place_name!=""
    placeLookup locationObject.place_name, placesApiKey, (placeData)->
      deferredLocation.resolve(placeData)
  else deferredLocation.resolve({error:'no place available'})
  deferredLocation.promise

# Main
go: (searchQuery, completeAction) ->
  placesApiKey = @placesApiKey
  fetchFromTwitter(searchQuery, @twitterApiKeys).then (twitterResults) ->
    promises = [] # array of google places promises
    for tweet in twitterResults
      promises.push findPlaceInfo tweet.location, placesApiKey
    q.all(promises).spread -> # When all the places promises have returned
      for tweet, index in twitterResults
        tweet.location = arguments[index] # Attach new location to Tweet
        tweet.sentiment = sentiment tweet.body # Attach sentiment to Tweet
        tweet.keywords = removeWords tweet.body # Attach keywords to Tweet
      completeAction twitterResults # Done!

module.exports = GetGeoSentimentTweets
```

6.5.4 utils/stream-handler.coffee

```
Tweet = require '../models/Tweet'

removeWords = require 'remove-words'
sentimentAnalysis = require 'sentiment-analysis'
moment = require 'moment'
placeLookup = require 'place-lookup'

placesApiKey = require('../config/keys').googlePlaces
blankPlace = { place_name: '', location: { lat: 0.000000, lng: 0.000000 } }

makeTweetObj = (data, location)->
  body: data.body
  dateTime: data.date
  keywords : removeWords(data.body)
  sentiment : sentimentAnalysis(data.body)
  location : location

# Determines if a Tweet object is complete & if it should be saved in the db
isSuitableForDb = (tweetData) ->
  if tweetData.sentiment == 0 then return false # Reject neutral tweets
  if tweetData.location.error? then return false # Reject no location tweets
  # Double check location is actually there, and reject if not
  if !tweetData.location.location.lat? then return false
  if !tweetData.location.location.lng? then return false
  # new! to make the db fill up more slowly, reject all slightly neutral tweets
  if tweetData.sentiment < 0.4 && tweetData.sentiment > -0.2 then return false
  # And reject tweets that out URL's - they're usually crap!
  if tweetData.body.indexOf('http') != -1 then return false
  return true

module.exports = (data, io) ->

  # Emit all tweets to the anyTweet listener
  anyTweet = makeTweetObj data, data.location
  anyTweet.dateTime = moment(new Date(anyTweet.dateTime)).fromNow()
  io.emit 'anyTweet', anyTweet

  if data.location.location.lat !=0
    tweet = makeTweetObj data, data.location
    io.emit 'tweet', tweet
    if isSuitableForDb tweet
      Tweet.findOneAndUpdate
        body: tweet.body,
        tweet,
        upsert: true,
        (err) -> if err then console.log 'ERROR UPDATING TWEET - '+err

  ## Uncomment this to stream live geo-accurate tweets- it will drain API usage
  # else if data.location.location.lat == 0 and data.location.place_name != ''
  #   placeLookup data.location.place_name, placesApiKey, (placeResults) ->
  #     tweetLocation = if !placeResults.error then placeResults else blankPlace
  #     tweet = makeTweetObj(data, tweetLocation)
  #
  #     if isSuitableForDb tweet
  #       Tweet.findOneAndUpdate
  #         body: tweet.body,
  #         tweet,
  #         upsert: true,
  #         (err) -> if err then console.log 'ERROR UPDATING TWEET - '+err
  #
  #     io.emit 'tweet', tweet # If everythinks cool, emit the tweet
```


6.5.5 utils/format-for-keyword-vis.coffee

```
Tweet = require './models/Tweet' # The Tweet model
MakeSummarySentences = require './utils/make-summary-sentences'
removeWords = require 'remove-words'
sentimentAnalysis = require 'sentiment-analysis'
FetchTweets = require 'fetch-tweets'

# API keys
twitterKey = require('./config/keys').twitter

fetchTweets = new FetchTweets twitterKey

class FormatWordsForCloud

  # Converts ordinary Tweet array to suitable form for word cloud
  formatResultsForCloud= (twitterResults, allWords = false) ->
    results = []
    tweetWords = makeTweetWords twitterResults
    for word in tweetWords
      sent = sentimentAnalysis word
      if allWords or sent != 0
        f = results.filter((item) -> item.text == word)
        if f.length == 0 then results.push(text: word, sentiment: sent, freq: 1)
        else for res in results then if res.text == word then res.freq++
    results

  findTopWords= (cloudWords) ->
    posData = cloudWords.filter (cw) -> cw.sentiment > 0
    negData = cloudWords.filter (cw) -> cw.sentiment < 0
    neutData = cloudWords.filter (cw) -> cw.sentiment == 0

    posData.sort (a, b) -> parseFloat(a.freq) - parseFloat(b.freq)
    posData = posData.reverse().slice(0,5)
    negData.sort (a, b) -> parseFloat(a.freq) - parseFloat(b.freq)
    negData = negData.reverse().slice(0,5)
    neutData.sort (a, b) -> parseFloat(a.freq) - parseFloat(b.freq)
    neutData = neutData.reverse().slice(0,5)

    {topPositive: posData, topNegative: negData, topNeutral: neutData}

  findTrendingWords = (cloudWords) ->
    cloudWords.sort (a, b) -> parseFloat(a.freq) - parseFloat(b.freq)
    cloudWords = cloudWords.reverse().slice(0,10)

  # Make a paragraph of keywords
  makeTweetWords = (twitterResults) ->
    para = ''
    for tweet in twitterResults then para += tweet.body + ' '
    removeWords para, false

  # Merge two sets of results
  mergeResults = (res1, res2) -> res1.concat res2

  # Make sentence description for map
  makeSentence = (data, searchTerm) ->
    (new MakeSummarySentences data, searchTerm).makeMapSentences()

  # Calls methods to fetch and format Tweets from the database
  renderWithDatabaseResults: (cb) ->
    Tweet.getAllTweets (tweets) ->
      cloudData = formatResultsForCloud(tweets)
      sentence = makeSentence(tweets, null)
      sentence.topWords = findTopWords cloudData
      cb cloudData, sentence
```

```

# Calls methods to fetch fresh Twitter, sentiment, and place data
renderWithFreshData: (searchTerm, cb) ->
  fetchTweets.byTopic searchTerm, (webTweets) ->
    Tweet.searchTweets searchTerm, (dbTweets) -> # Fetch matching db results
      data = mergeResults webTweets, dbTweets
      cloudData = formatResultsForCloud(data, true)
      sentence = makeSentence(data, searchTerm)
      sentence.topWords = findTopWords cloudData
      sentence.trending = findTrendingWords cloudData
      cb cloudData, sentence

findTrends: (tweets) ->
  findTopWords formatResultsForCloud tweets, true

fwfc = new FormatWordsForCloud()
module.exports.getFreshData = fwfc.renderWithFreshData
module.exports.getDbData = fwfc.renderWithDatabaseResults
module.exports.findTopWords = fwfc.findTrends

```

6.5.6 utils/format-tweets-for-map.coffee

```

Tweet = require '../models/Tweet' # The Tweet model
CompleteTweets = require '../utils/get-complete-tweets' # Fetches & formats data
MakeSummarySentences = require '../utils/make-summary-sentences'

```

```

# API keys
twitterKey = require('../config/keys').twitter
googlePlacesKey = require('../config/keys').googlePlaces

```

```

class FormatTweetsForMap

```

```

  # Slightly blur location data, as to not reveal users exact position
  blurLocationData = (loc) ->
    loc = loc + '' # Convert location part into a string
    accuracy = 3 # -1 = 110km, 1 = 10km, 2 = 1km, 3 = 100m, 4 = 10m ...
    digitIndex = (loc).indexOf('.') + accuracy # Find index of digit to modify
    randomDigit = Math.floor(Math.random() * 10)
    if loc.length > digitIndex
      loc = loc.substr(0, digitIndex) + randomDigit + loc.substr(digitIndex + 1)
    Number loc # Convert result back to a number, and return

  # Converts ordinary Tweet array to lat + lng array for the heat map
  formatResultsForMap = (twitterResults) ->
    mapData = []
    for tweet in twitterResults
      if !tweet.location.error?
        mapData.push
          sentiment: tweet.sentiment
          location:
            lat: blurLocationData tweet.location.location.lat
            lng: blurLocationData tweet.location.location.lng
          tweet: tweet.body
    mapData

  # Inserts an array of valid Tweets into the database, if not already
  insertTweetsIntoDatabase = (twitterResults) ->
    for tweet in twitterResults
      tweetData =
        body: tweet.body
        dateTime: tweet.date
        sentiment : tweet.sentiment
        location : tweet.location
      if isSuitableForDb tweetData
        Tweet.findOneAndUpdate
          body: tweetData.body,

```

```

    tweetData,
    upsert: true,
    (err) -> if err then console.log 'ERROR UPDATING TWEET - ' + err

# Determines if a Tweet object is complete & if it should be saved in the db
isSuitableForDb = (tweetData) ->
  if tweetData.sentiment == 0 then return false
  if tweetData.location.error? then return false
  if !tweetData.location.location.lat? then return false
  if !tweetData.location.location.lng? then return false
  return true

# Merge two sets of results
mergeResults = (res1, res2) -> res1.concat res2

# Make sentence description for map
makeSentence = (data, searchTerm) ->
  (new MakeSummarySentences data, searchTerm).makeMapSentences()

# Calls methods to fetch and format Tweets from the database
renderWithDatabaseResults: (cb) ->
  Tweet.getAllTweets (tweets) ->
    cb formatResultsForMap(tweets), makeSentence(tweets, null)

# Calls methods to fetch fresh Twitter, sentiment, and place data
renderWithFreshData: (searchTerm, cb) ->
  completeTweets = new CompleteTweets(twitterKey, googlePlacesKey)
  completeTweets.go searchTerm, (webTweets) ->
    insertTweetsIntoDatabase(webTweets) # Add new Tweets to the db
    Tweet.searchTweets searchTerm, (dbTweets) -> # Fetch matching db results
      data = mergeResults webTweets, dbTweets
      cb formatResultsForMap(data), makeSentence(data, searchTerm)

ftfm = new FormatTweetsForMap()
module.exports.getFreshData = ftfm.renderWithFreshData
module.exports.getDbData = ftfm.renderWithDatabaseResults

```

6.5.7 utils/make-click-words.coffee

```
removeWords = require 'remove-words'

# Makes keywords click-able hyperlinks, returns HTML
makeClickWords = (body) ->
  clWord = (word) -> (''+word).toLowerCase().replace /\W/g, ''
  clickWords = removeWords body # Array of keywords
  htmlTweet = ''
  aStyle = 'style="color: black; font-weight: bold;" ' # style for hyperlinks
  for word in body.split " "
    if clWord(word) in clickWords
      htmlTweet += "<a #{aStyle} href='/text-tweets/#{clWord word}'>#{word}</a> "
    else htmlTweet += "#{word} "
  htmlTweet

module.exports = makeClickWords
```

6.5.8 utils/Watson-tone-analyser.coffee

```
watson = require('watson-developer-cloud')
watsonCredentials = require('../config/keys').watson

# Format tone data
formatResults = (toneResults) ->
  results = []
  toneCategories = toneResults.document.tone.tone_categories
  for toneCategory in toneCategories.slice(0,toneCategories.length-1)
    for tone in toneCategory.tones
      results.push {name: tone.tone_name, score: tone.score}
  results

# Fetch tone analyser results from BlueMix instance
fetchToneAnalyzerResults = (tweetsArr, callback) ->
  text = ''
  for tweet in tweetsArr then text += tweet.body + ' '
  tone_analyzer = watson.tone_analyzer(
    username: watsonCredentials.username
    password: watsonCredentials.password
    version: 'v3-beta'
    version_date: '2016-02-11')
  tone_analyzer.tone { text: text }, (err, tone) ->
    if err then console.log err; callback {}
    else callback formatResults tone

module.exports = fetchToneAnalyzerResults
```

6.5.9 utils/make-summary-sentence.coffee

```
class MakeSummarySentences

  constructor: (@tweetObjects, @searchTerm = null) ->

  # Finds the average value for an array of numbers
  findAv = (arr) ->
    t = 0
    for i in arr then t += i
    t/arr.length

  # Finds average positive, negative and overall sentiment from list of tweets
  getAverageSentiments = (tweetObjects) ->
    posSent = []
    negSent = []
    neuSent = []

    for item in tweetObjects
      if item.sentiment > 0 then posSent.push item.sentiment
      else if item.sentiment < 0 then negSent.push item.sentiment
      else neuSent.push(0)

    avPositive: Math.round(findAv(posSent) * 100)
    avNegative: Math.round(findAv(negSent) * -100)
    avSentiment: Math.round(findAv(posSent.concat(negSent).concat(neuSent)) * 100)
    totalPositive: posSent.length
    totalNegative: negSent.length

  # Determines if the overall sentiment is "Positive", "Negative" or "Neutral"
  getOverallSentimentName = (avSentiment) ->
    if avSentiment > 0 then "Positive"
    else if avSentiment < 0 then "Negative"
    else "Neutral"

  makeTxtStyle = (sentiment) ->
    col =
      if sentiment > 0 or sentiment == 'Positive' then 'green'
      else if sentiment < 0 or sentiment == 'Negative' then 'darkred'
      else 'gray'
    " style='font-weight: bold; color: #{col}' "

  makeGlobeSentence = (tweetObjects, relTo, averages, overallSent) ->
    numRes = "<b><span id='numRes'>#{tweetObjects.length}</span></b>"
    overallSentTxt = "<span #{makeTxtStyle overallSent} >#{overallSent}"
    overallSentTxt += " (#{averages.avSentiment}%)</span>"
    positivePercent = "<span #{makeTxtStyle 1}>#{averages.avPositive}</span>"
    negativePercent = "<span #{makeTxtStyle -1}>#{averages.avNegative}</span>"
    s = "Globe displaying #{numRes} sentiment values calculated "
    s += "from Twitter results #{relTo} "
    s += "the overall sentiment is #{overallSentTxt}."
    s += "<br><br>"
    s += " #{averages.totalPositive} Tweets are positive "
    s += "with an average sentiment of #{positivePercent} "
    s += "and #{averages.totalNegative} Tweets are negative "
    s += "with an average sentiment of #{negativePercent}."
    s

  makeMapSentences = (tweetObjects, averages, overallSent, relTo) ->
    numRes = "<b><span id='numRes'>#{tweetObjects.length}</span></b>"
    mapShowing = "Map showing #{numRes} "
    mapShowing += "of the latest Twitter results #{relTo}<br>"
    mapShowing += "Overall sentiment is: "
    mapShowing += "<span #{makeTxtStyle overallSent} >#{overallSent} "
    mapShowing += " (#{averages.avSentiment}%)</span>"

    p = makeTxtStyle 1
    n = makeTxtStyle -1
```

```

sentimentSummary="Average positive: <b # {p}>#{averages.avPositive}%</b>.<br>"
sentimentSummary+="Average negative: <b # {n}>#{averages.avNegative}%</b>."

mapShowing: mapShowing
sentimentSummary: sentimentSummary

# Makes the sentences for the map
makeMapSentences: () ->
  averages = getAverageSentiments(@tweetObjects)
  overallSent = getOverallSentimentName(averages.avSentiment)
  relTo = if @searchTerm? then "relating to <b>#{@searchTerm}</b>" else ""

  mapSentences = makeMapSentences @tweetObjects, averages, overallSent, relTo
  mapShowing: mapSentences.mapShowing
  sentimentSummary: mapSentences.sentimentSummary
  globeSentence: makeGlobeSentence @tweetObjects, relTo, averages, overallSent
  searchTerm: if @searchTerm then @searchTerm else ''

module.exports = MakeSummarySentences

```

6.5.10 utils/make-timeline-data.coffee

```

class MakeSummarySentences

  constructor: (@tweetObjects, @searchTerm = null) ->

  # Finds the average value for an array of numbers
  findAv = (arr) ->
    t = 0
    for i in arr then t += i
    t/arr.length

  # Finds average positive, negative and overall sentiment from list of tweets
  getAverageSentiments = (tweetObjects) ->
    posSent = []
    negSent = []
    neuSent = []

    for item in tweetObjects
      if item.sentiment > 0 then posSent.push item.sentiment
      else if item.sentiment < 0 then negSent.push item.sentiment
      else neuSent.push(0)

    avPositive: Math.round(findAv(posSent) * 100)
    avNegative: Math.round(findAv(negSent) * -100)
    avSentiment: Math.round(findAv(posSent.concat(negSent).concat(neuSent)) * 100)
    totalPositive: posSent.length
    totalNegative: negSent.length

  # Determines if the overall sentiment is "Positive", "Negative" or "Neutral"
  getOverallSentimentName = (avSentiment) ->
    if avSentiment > 0 then "Positive"
    else if avSentiment < 0 then "Negative"
    else "Neutral"

  makeTxtStyle = (sentiment) ->
    col =
      if sentiment > 0 or sentiment == 'Positive' then 'green'

```

```

    else if sentiment < 0 or sentiment == 'Negative' then 'darkred'
    else 'gray'
" style='font-weight: bold; color: #{col}' "

makeGlobeSentence = (tweetObjects, relTo, averages, overallSent) ->
  numRes = "<b><span id='numRes'>#{tweetObjects.length}</span></b>"
  overallSentTxt = "<span #{makeTxtStyle overallSent} >#{overallSent}"
  overallSentTxt += " (#{averages.avSentiment}%)</span>"
  positivePercent = "<span #{makeTxtStyle 1}>#{averages.avPositive}%</span>"
  negativePercent = "<span #{makeTxtStyle -1}>#{averages.avNegative}%</span>"
  s = "Globe displaying #{numRes} sentiment values calculated "
  s += "from Twitter results #{relTo} "
  s += "the overall sentiment is #{overallSentTxt}."
  s += "<br><br>"
  s += "#{averages.totalPositive} Tweets are positive "
  s += "with an average sentiment of #{positivePercent} "
  s += "and #{averages.totalNegative} Tweets are negative "
  s += "with an average sentiment of #{negativePercent}."
  s

makeMapSentences = (tweetObjects, averages, overallSent, relTo) ->
  numRes = "<b><span id='numRes'>#{tweetObjects.length}</span></b>"
  mapShowing = "Map showing #{numRes} "
  mapShowing += "of the latest Twitter results #{relTo}<br>"
  mapShowing += "Overall sentiment is: "
  mapShowing += "<span #{makeTxtStyle overallSent} >#{overallSent} "
  mapShowing += " (#{averages.avSentiment}%)</span>"

  p = makeTxtStyle 1
  n = makeTxtStyle -1

  sentimentSummary="Average positive: <b # {p}>#{averages.avPositive}%</b>.<br>"
  sentimentSummary+="Average negative: <b # {n}>#{averages.avNegative}%</b>."

  mapShowing: mapShowing
  sentimentSummary: sentimentSummary

# Makes the sentences for the map
makeMapSentences: () ->
  averages = getAverageSentiments(@tweetObjects)
  overallSent = getOverallSentimentName(averages.avSentiment)
  relTo = if @searchTerm? then "relating to <b>#{@searchTerm}</b>" else ""

  mapSentences = makeMapSentences @tweetObjects, averages, overallSent, relTo
  mapShowing: mapSentences.mapShowing
  sentimentSummary: mapSentences.sentimentSummary
  globeSentence: makeGlobeSentence @tweetObjects, relTo, averages, overallSent
  searchTerm: if @searchTerm then @searchTerm else ''

module.exports = MakeSummarySentences

```

6.6 MODELS

6.6.1 models/Tweet.coffee

```
mongoose = require 'mongoose'

schema = new mongoose.Schema({
  body      : String
  dateTime  : String
  sentiment : Number
  location  : { type : Object , "default" : {} }
}),
{ capped: { size: 491520, max: 1500, autoIndexId: true } }

schema.statics.getAllTweets = (callback) ->
  Tweet.find {}, (err, results) ->
    callback results

schema.statics.searchTweets = (searchTerm, callback) ->
  Tweet.find { keywords: searchTerm }, (err, results) ->
    callback results

module.exports = Tweet = mongoose.model('Tweet', schema)
```


6.7 API-ROUTES

6.7.1 api-routes/db-api.coffee

```
express = require('express')
router = express.Router()
fetchSentimentTweets = require '../utils/fetch-sentiment-tweets'

router.post '/', (req, res) ->
  fetchSentimentTweets '', (data, average) ->
    res.json data

module.exports = router
```

6.7.2 api-routes/antity-api.coffee

```
express = require('express')
router = express.Router()

entityExtraction = require 'haven-entity-extraction'
hpKey = require('../config/keys').hp

# Formats tweets into a massive tweet body
formatText = (tweetBody) ->
  tweetBody = tweetBody.replace(/(?:https?|ftp):\/\/[\n\S]+/g, '')
  tweetBody = tweetBody.replace(/[^A-Za-z0-9 ]/g, '')
  tweetBody = tweetBody.substring(0, 5000)

formatData = (data) ->
  results = []
  for key in Object.keys data
    category = key.charAt(0).toUpperCase()+key.split('_')[0].slice(1)
    results.push name: category, items: data[key]
  results = results.sort (a, b) -> b.items.length - a.items.length

router.post '/', (req, res) ->
  entityExtraction formatText(req.body.text), hpKey, (data) ->
    res.json formatData data

module.exports = router
```

6.7.3 api-routes/tone-api.coffee

```
express = require('express')
router = express.Router()
fetchSentimentTweets = require '../utils/fetch-sentiment-tweets'

watson = require 'watson-developer-cloud'
watsonCredentials = require('../config/keys').watson

toneAnalyzer = watson.tone_analyzer {
  url: 'https://gateway.watsonplatform.net/tone-analyzer-beta/api/'
  username: watsonCredentials.username
  password: watsonCredentials.password
  version_date: '2016-11-02'
  version: 'v3-beta'
}

noBodyProvided = (req, res, next) ->
  makeBody = (twAr) ->
    tweetBody = ''
    for tw in twAr then tweetBody += tw.body + ' '
    tweetBody = tweetBody.replace(/(?:https?|ftp):\/\/[^\n\S]+/g, '')
    tweetBody = tweetBody.replace(/[^A-Za-z0-9 ]/g, '').substring(0, 5000)

  toneRes = (body) ->
    toneAnalyzer.tone {text: body}, (err, data) ->
      if err then next err else res.json data

  if req.body.searchTerm? && req.body.searchTerm != ''
    fetchSentimentTweets req.body.searchTerm, (r) -> toneRes makeBody r
  else fetchSentimentTweets '', (r) -> toneRes makeBody r

router.post '/', (req, res, next) ->
  if !req.body.text? or req.body.text == '' then noBodyProvided req, res, next
  else
    toneAnalyzer.tone req.body, (err, data) ->
      if err then next err else res.json data

module.exports = router
```

6.7.4 api-routes/trending-api.coffee

```
express = require('express')
router = express.Router()
FetchTweets = require 'fetch-tweets'
twitterKey = require('../config/keys').twitter
asyncTweets = require '../utils/async-tweets'
placeLookup = require 'place-lookup'
placesKey = require('../config/keys').googlePlaces

fetchTweets = new FetchTweets twitterKey

findAverageSentiment = (tweetArr) ->
  totalSentiment = 0
  for tweet in tweetArr.tweets then totalSentiment += tweet.sentiment
  totalSentiment / tweetArr.tweets.length

findOriginalTrend = (searchTerm, searchTerms, trends) ->
  for st, i in searchTerms then if st == searchTerm then return trends[i]
  {}

makeTrendRequest = (woeid, cb) ->
  fetchTweets.trending woeid, (trendingResults) ->

  # Make a list of searchTerms from the trending topics
  searchTerms = []
  originalTrends = []
  i = 0
  while searchTerms.length < 10
    if !trendingResults[i]? then cb {}; return
    trend = trendingResults[i].trend.replace(/[\\W_]+/g, '')
    if trend.length > 0
      searchTerms.push trend
      originalTrends.push trendingResults[i]
    i++

  # Make the actual request
  asyncTweets searchTerms, (twitterResults) ->

  #Make the results array
  results = []
  for tweetArr in twitterResults
    t = findOriginalTrend(tweetArr.searchTerm, searchTerms, trendingResults)
    results.push
      topic: t.trend
      sentiment: findAverageSentiment tweetArr
      volume: t.volume
  cb results

router.post '/', (req, res) ->
  woeid = 1
  makeTrendRequest woeid, (results) -> res.json {trends: results}

router.post '/:location', (req, res) ->
  fuzzyLocation = req.params.location
  placeLookup fuzzyLocation, placesKey, (placeResults) ->
    if !placeResults.error?
      lat = placeResults.location.lat
      lng = placeResults.location.lng
      fetchTweets.closestTrendingWoeid lat, lng, (places) ->
        woeid = places[0].woeid
        makeTrendRequest woeid, (results) ->
          if results == {} then res.json {}
          res.json {trends: results, location: placeResults.place_name}
    else res.json {}
module.exports = router
```

6.8 ROUTES

6.8.1 routes/about.coffee

```
express = require('express')
router = express.Router()
router.get '/', (req, res, next) ->
  res.render 'page_about',
    title: 'About'
    pageNum: -1
module.exports = router
```

6.8.2 routes/index.coffee

```
# Require necessary modules, API keys and instantiate objects
fetchSentimentTweets = require '../utils/fetch-sentiment-tweets'
express = require('express')
router = express.Router()

# Main route - no search term
router.get '/', (req, res, next) ->
  fetchSentimentTweets '', (results, average) -> # Fetch all data
    res.render 'index', # Render template
      title: 'Sentiment Sweep'
      pageNum: 0
      data: results
      averageSentiment: average

module.exports = router
```

6.8.3 routes/now.coffee

```
express = require('express')
router = express.Router()

router.get '/', (req, res) ->
  res.render 'page_now',
    title: 'The world right now | Sentiment Sweep'
    pageNum: 15

module.exports = router
```

6.8.4 routes/real-time-dash.coffee

```
express = require('express')
router = express.Router()

router.get '/', (req, res) ->
  res.render 'page_realtime',
    title: 'Real-time Dashboard | Sentiment Sweep'
    pageNum: 13

module.exports = router
```

6.8.5 routes/break-down.coffee

```
# Require necessary modules, API keys and instantiate objects
Tweet = require './models/Tweet' # The Tweet model
moment = require 'moment'
FetchTweets = require 'fetch-tweets'
twitterKey = require('./config/keys').twitter
fetchTweets = new FetchTweets twitterKey
removeWords = require 'remove-words'
hpSentimentAnalysis = require 'haven-sentiment-analysis'
hpKey = require('./config/keys').hp
express = require('express')
router = express.Router()

fetchAndFormatTweets = (searchTerm, cb) ->
  fetchTweets.byTopic searchTerm, (results) ->
    cb formatTweets results

formatTweets = (twitterResults) ->
  results = ""
  for tweet in twitterResults then results += tweet.body + " "
  results = results.replace(/(?:https?|ftp):\/\/[^\n\S]+/g, '')
  results = results.replace(/[^A-Za-z0-9 ]/g, '')
  results = results.substring(0, 5000)

getHpSentimentResults = (tweetBody, cb) ->
  hpSentimentAnalysis tweetBody, hpKey, (results) ->
    cb results

formatResultsForChart = (hpResults) ->
  data = {
    name: 'sentiment-tree', children: [
      {name: 'positive', children: []}
      {name: 'negative', children: []}
    ]
  }
  i = 1
  while i <= 10
    data.children[0].children.push({name: i/10, children: [] })
    data.children[1].children.push({name: i/-10+'', children: [] })
    i++

  for posRes in hpResults.positive
    score = Math.round(posRes.score*10)/10
    for i in data.children[0].children
      if i.name == score then i.children.push(
        name: posRes.sentiment
        size: posRes.score
        topic: posRes.topic
      )

  for negRes in hpResults.negative
    score = Math.round(negRes.score*10)/10+' '
    for i, index in data.children[1].children
      if i.name == score then i.children.push(
        name: negRes.sentiment
        size: Math.abs(negRes.score)
        topic: negRes.topic
      )
  data

# Route with search term
router.get '/:query', (req, res) ->
  searchTerm = req.params.query # Get the search term from URL param
```

```

fetchAndFormatTweets searchTerm, (tweetBody) ->
  getHpSentimentResults tweetBody, (hpResults) ->
    results = formatResultsForChart hpResults
    res.render 'page_breakDown',
      title: searchTerm+' results'
      pageNum: -1
      data: results
      searchTerm: searchTerm

router.get '/', (req, res) ->
  Tweet.getAllTweets (tweets) ->
    tweetBody = formatTweets tweets
    getHpSentimentResults tweetBody, (hpResults) ->
      results = formatResultsForChart hpResults
      res.render 'page_breakDown',
        title: 'Break down results'
        pageNum: -1
        data: results
        searchTerm: ''

module.exports = router

```

6.8.6 routes/timeline.coffee

```

express = require('express')
router = express.Router()

tweetTimeFormatter = require './utils/make-timeline-data'

router.get '/', (req, res, next) ->
  tweetTimeFormatter.getDbData (data, txt) ->
    res.render 'page_timeline',
      title: 'Time Line'
      pageNum: 3
      data: data
      searchTerm: ''

router.get '/:query', (req, res) ->
  searchTerm = req.params.query # Get the search term from URL param
  tweetTimeFormatter.getFreshData searchTerm, (data, txt) ->
    res.render 'page_timeline',
      title: 'Time Line'
      pageNum: 3
      data: data
      searchTerm: searchTerm

module.exports = router

```

6.8.7 routes/region-map.coffee

```
fs = require 'fs'
express = require('express')
router = express.Router()

findRegion = require 'find-region-from-location'

mapTweetFormatter = require '../utils/format-tweets-for-map'

makeRegionMapData = (tweets) ->

  # Finds the average value for an array of numbers
  findAv = (arr) ->
    t = 0
    for i in arr then t += i
    Math.round(t/arr.length*100)/100

  # Group together the sentiments to their regions
  prelimResults = {}
  for tweet in tweets
    region = findRegion.country(tweet.location.lat, tweet.location.lng)
    if prelimResults[region] then
prelimResults[region].sentiments.push(tweet.sentiment)
    else prelimResults[region] = {region: region, sentiments: [tweet.sentiment]}

  # Make the results array, by finding the average sentiment for each region
  results = []
  for regionKey of prelimResults
    if prelimResults.hasOwnProperty regionKey
      results.push([regionKey, findAv(prelimResults[regionKey].sentiments)])

  # Order results by sentiment, then append column headings and min/max values
  sortFunc = (a, b) -> if a[1] == b[1] then 0 else if a[1] < b[1] then -1 else 1
  results.sort sortFunc
  results.unshift ['Country', 'Sentiment'], ['', -0.8], ['', 0.8]
  results

getRegions = () ->
  fs.readFileSync(__dirname+'../public/data/regions.csv','utf8').split('\r\n')

# Render to page
render = (res, data, title, summaryTxt, location = '') ->
  summaryTxt.searchRegion = location
  res.render 'page_regions', # Call res.render for the map page
  data: data # The map data
  summary_text: summaryTxt # Summary of results
  title: title # The title of the rendered map
  pageNum: 6 # The position in the application
  csvRegions: getRegions() # List of all regions

# Call render with search term
renderSearchTerm = (res, searchTerm, location= '') ->
  mapTweetFormatter.getFreshData searchTerm, (twitterData, summaryTxt) ->
    regionData = makeRegionMapData twitterData
    render res, regionData, searchTerm+' Region Map', summaryTxt, location

# Call render for database data
renderAllData = (res, location= '') ->
  mapTweetFormatter.getDbData (data, txt) ->
    render res, makeRegionMapData(data), 'Map', txt, location

# Path for main map root page
router.get '/', (req, res) -> renderAllData res

# Path for map sub-page
router.get '/:query', (req, res) ->
  searchTerm = req.params.query # Get the search term from URL param
```

```

    if searchTerm != null then renderSearchTerm res, searchTerm
    else renderAllData res

# Path for map location sub-page
router.get '/location/:query', (req, res) ->
  location = req.params.query # Get the location from URL param
  renderAllData res, location

module.exports = router

```

6.8.8 routes/tone-analyzer.coffee

```

express = require('express')
router = express.Router()
router.get '/', (req, res, next) ->
  res.render 'page_tones',
    title: 'Tone Analyzer'
    searchTerm: ''
    pageNum: 9

router.get '/:query', (req, res, next) ->
  res.render 'page_tones',
    title: 'Tone Analyzer'
    searchTerm: req.params.query
    pageNum: 9
module.exports = router

```

6.8.9 routes/trending.coffee

```

express = require('express')
router = express.Router()

router.get '/', (req, res, next) ->
  res.render 'page_trending',
    title: 'Trending Now'
    pageNum: 11
    location: ''

router.get '/:query', (req, res) ->
  location = req.params.query # Get the search term from URL param
  res.render 'page_trending',
    title: 'Trending in '+location
    pageNum: 11
    location: location

module.exports = router

```


6.8.10 routes/sa-comparison.coffee

```
express = require('express')
router = express.Router()

defaultRes = require('../public/data/all_sentiment_results_edwardsnowdon.json')

# Renders the layout with the sentiment results passed as param
renderWithResults = (res, results, searchTerm) ->
  res.render('page_comparison', {results: results, searchTerm: searchTerm})
# res.json results

# Root task, called when no params passed, should use default results
router.get '/', (req, res, next) ->
  renderWithResults res, defaultRes, ''

# Fetches Tweets for given query, calculates sentiment then calls render
router.get('/:query', (req, res, next) ->

  # Node modules
  fetchTweets = require('fetch-tweets')
  dictionarySA = require('sentiment-analysis')
  nluSA = require('haven-sentiment-analysis')

  # Keys and instance variables
  hpKey = require('../config/keys').hp
  twitterKey = require('../config/keys').twitter
  results = [] # Will hold list of json objects to be rendered
  completedRequests = 0 # Counts how many results returned
  searchTerm = req.params.query # Get the search term from URL param

  # Method will generate an object with all calculated sentiments for tweet
  calculateResults = (searchTerm) ->
    makeSentimentResults = (rawTweets) ->
      rawTweets.forEach (tweet, index) ->
        nluSA { text: tweet }, hpKey, (nluResults) ->
          results.push
            index: index
            tweet: tweet
            dictionary_sentiment: dictionarySA(tweet)
            nlu_sentiment: Math.round(nluResults.aggregate.score * 1000) / 1000
            human_sentiment: null
          completedRequests++
        if completedRequests == rawTweets.length # Everything is done, render
          renderWithResults res, results, searchTerm

  # Fetch the Tweets
  fetchTweets = new fetchTweets(twitterKey)
  searchOptions =
    q: searchTerm
    lang: 'en'
    count: 50
  fetchTweets.byTopic searchOptions, (results) ->
    rawTweets = []
    results.forEach (tweet) -> rawTweets.push tweet.body
    makeSentimentResults rawTweets

  if searchTerm != null
    calculateResults searchTerm
  else
    renderWithResults res, defaultRes, ''

module.exports = router
```

6.8.11 routes/search.coffee

```
# Require necessary modules, API keys and instantiate objects
fetchSentimentTweets = require './utils/fetch-sentiment-tweets'
wordFormatter = require('./utils/format-for-keyword-vis').findTopWords
#toneAnalyzer = require './utils/watson-tone-analyzer'
makeClickWords = require './utils/make-click-words'

express = require('express')
router = express.Router()

# Converts Tweet objects into the right format
formatResults = (tweetArr) ->
  results = {}
  # Add keywords list
  wrdsjs = wordFormatter(tweetArr, true)
  topData = wrdsjs.topPositive.concat(wrdsjs.topNegative, wrdsjs.topNeutral)
  topData = topData.sort (a, b) -> parseFloat(b.freq) - parseFloat(a.freq)
  topData = topData.splice(0,10)
  results.keywordData= topData = topData.sort -> 0.5 - Math.random()

  # Find average sentiment
  totalSentiment = 0
  for tweet in tweetArr then totalSentiment += tweet.sentiment
  results.averageSentiment = totalSentiment / tweetArr.length

  # Find percentage positive, negative and neutral
  pieChart = {positive: 0, neutral: 0, negative: 0}
  for tweet in tweetArr
    if tweet.sentiment > 0 then pieChart.positive += 1
    else if tweet.sentiment < 0 then pieChart.negative += 1
    else pieChart.neutral += 1
  results.pieChart = pieChart
  results.tweets = tweetArr
  results

getTopTweets = (tweetArr) ->
  tweetArr = JSON.parse(JSON.stringify(tweetArr)) # Clone tweetArr
  results = {}
  topPositiveTweets = tweetArr.sort((b, a) ->
    a.sentiment - (b.sentiment)
  ).slice(0, 5)
  topNegativeTweets = tweetArr.sort((a, b) ->
    a.sentiment - (b.sentiment)
  ).slice(0, 5)
  for tweet in topPositiveTweets then tweet.body = makeClickWords tweet.body
  for tweet in topNegativeTweets then tweet.body = makeClickWords tweet.body
  results.topPositive = topPositiveTweets
  results.topNegative = topNegativeTweets
  results

makeTweetBody = (tweetArr) ->
  results = ""
  for tweet in tweetArr then results += tweet.body + ' '
  results

# Main route - no search term
router.get '/', (req, res, next) ->
  res.render 'page_search',
    title: 'Search'
    pageNum: -1
    data: {}
    searchTerm: ''

# Route with search term
router.get '/:query', (req, res) ->
```

```

searchTerm = req.params.query # Get the search term from URL param

fetchSentimentTweets searchTerm, (results, average) -> # Fetch all data

res.render 'page_search', # Render template
  title: searchTerm+' results'
  pageNum: -1
  data: formatResults results
  averageSentiment: average
  searchTerm: searchTerm
  topTweets: getTopTweets results
  tweetBody: makeTweetBody results

module.exports = router

```

6.8.12 routes/word-cloud.coffee

```

express = require('express')
router = express.Router()

tweetWordFormatter = require '../utils/format-for-keyword-vis'

router.get '/', (req, res, next) ->
  tweetWordFormatter.getDbData (data, txt) ->
    res.render 'page_cloud',
      title: 'Word Cloud'
      pageNum: 5
      summary_text: txt
      data: data

router.get('/:query', (req, res) ->
  searchTerm = req.params.query # Get the search term from URL param
  tweetWordFormatter.getFreshData searchTerm, (data, txt) ->
    res.render 'page_cloud',
      title: 'Word Cloud'
      pageNum: 5
      summary_text: txt
      data: data

module.exports = router

```

6.8.13 routes/text-tweets.coffee

```
# Require necessary modules, API keys and instantiate objects
Tweet = require '../models/Tweet' # The Tweet model
sentimentAnalysis = require 'sentiment-analysis'
moment = require 'moment'
makeClickWords = require '../utils/make-click-words'
FetchTweets = require 'fetch-tweets'
twitterKey = require('../config/keys').twitter
fetchTweets = new FetchTweets twitterKey
removeWords = require 'remove-words'
express = require('express')
router = express.Router()

# Converts Tweet objects into the right format
formatTweets = (tweets) ->
  pos = []
  neg = []

  tweets.sort (b, a) -> new Date(a.dateTime).getTime() - new
Date(b.dateTime).getTime()
  tweets.slice(0,350)

  for t in tweets
    body = makeClickWords t.body
    location = t.location.place_name
    sentiment = if t.sentiment then t.sentiment else sentimentAnalysis t.body
    keywords = removeWords t.body
    r = body: body, location: location, sentiment: sentiment, keywords: keywords
    r.dateTime = moment(new Date(t.dateTime)).fromNow()
    if sentiment > 0 then pos.push r else if sentiment < 0 then neg.push r

  pos.sort (b, a) -> parseFloat(a.sentiment) - parseFloat(b.sentiment)
  neg.sort (a, b) -> parseFloat(a.sentiment) - parseFloat(b.sentiment)
  positive: pos.slice(0,100), negative: neg.slice(0,100)

# Main route - no search term
router.get '/', (req, res, next) ->
  Tweet.getAllTweets (tweets) ->
    res.render 'page_textTweets',
      title: 'View Raw Tweets'
      pageNum: 7
      data: formatTweets tweets
      searchTerm: ''

# Route with search term
router.get '/:query', (req, res) ->
  searchTerm = req.params.query # Get the search term from URL param
  fetchTweets.byTopic searchTerm, (tweets) ->
    res.render 'page_textTweets',
      title: searchTerm+' raw tweets'
      pageNum: 7
      data: formatTweets tweets
      searchTerm: searchTerm

module.exports = router
```

6.8.14 routes/word-plot.coffee

```
express = require('express')
router = express.Router()

tweetWordFormatter = require './utils/format-for-keyword-vis'

router.get '/', (req, res, next) ->
  tweetWordFormatter.getDbData (data, txt) ->
    res.render 'page_wordPlot',
      title: 'Word Scatter Plot'
      pageNum: 5
      summary_text: txt
      data: data

router.get '/:query', (req, res) ->
  searchTerm = req.params.query # Get the search term from URL param
  tweetWordFormatter.getFreshData searchTerm, (data, txt) ->
    res.render 'page_wordPlot',
      title: 'Word Scatter Plot'
      pageNum: 5
      summary_text: txt
      data: data

module.exports = router
```

6.9 CONFIG

6.9.1 config/app-config.coffee

```
#Set the current environment to true in the env object
currentEnv = process.env.NODE_ENV or 'development'
exports.appName = "Twitter Sentiment Visualisations"
exports.env =
  production: false
  staging: false
  test: false
  development: false
exports.env[currentEnv] = true
exports.log =
  path: __dirname + "reports/log/app_#{currentEnv}.log"
exports.server =
  port: 8080
#In staging and production, listen loopback. nginx listens on the network.
  ip: '127.0.0.1'
if currentEnv not in ['production', 'staging']
  exports.enableTests = true
  #Listen on all IPs in dev/test (for testing from other machines)
  exports.server.ip = '0.0.0.0'
exports.db =
  URL: "mongodb://localhost/" +
    "#{exports.appName.toLowerCase()}_#{currentEnv}".replace RegExp(' ', 'g'), '-'
```

6.9.2 config/api-keys.coffee

```
exports.twitter =
  consumer_key : 'P7vhOiDbJWwNbEAb4OAKjNvvA'
  consumer_secret : 'OJh4gToZW2P37011GRP7zwChyAbAnzUF5teck0hnQzqvaPlj8M'
  token: '43356339-APKxVQa9JJXfioSebRt8kQB9A5HcTf9kZvuz2vVdR'
  token_secret: 'ZpiQ9acPV4vaHYjNxzLGjhgSgme71v8LZJ8yxphYFu4XZ'

exports.googlePlaces = 'AIzaSyBLcuWU22gEmEzh9wiYbcBb1I6wGghRAKU'

exports.hp = 'bc3d211e-4212-417c-ae57-04523565c46a'

exports.watson =
  username: 'd640965f-8baa-4bc9-9e49-0f85a9f694e9'
  password: 'mP0Ie7HBgxbx'
```

6.10 CLIENT SIDE SCRIPTS

6.10.1 map/map-main.coffee

```
initialize = ->

  # Include all map components
  themeModule = require('../map/theme-module.coffee')
  optionsModule = require('../map/options-module.coffee')
  heatmapModule = require('../map/heatmap-module.coffee')
  searchModule = require('../map/search-module.coffee')
  autocompleteModule = require('../map/autocomplete-module.coffee')
  markerClusterModule = require('../map/markercluster-module.coffee')
  liveInteractions = require('../map/live-interactions-module.coffee')
  socketModule = require('../map/socket-module.coffee')

  # Get the map, and set the map options
  map = new (google.maps.Map) (document.getElementById('map-canvas'), \
    optionsModule.mapOptions)

  # Apply map theme
  map.mapTypes.set 'map_style', themeModule.styledMap
  map.setMapTypeId 'map_style'

  # Apply heat map
  heatmapModule.positiveHeatmap.setMap map
  heatmapModule.negativeHeatmap.setMap map
  heatmapModule.neutralHeatmap.setMap map

  # Apply invisible marker cluster to map, so user can click
  markerClusterModule.map

  # Initiate the places auto-complete and map search
  searchModule.initiatePlaceSearch map

  data = [
    { 'value': '11', 'label': 'one' }
    { 'value': '2', 'label': 'two' }
    { 'value': '3', 'label': 'three' }
  ]

  $(document).ready ->
    goToUrl = (url) -> window.location = url # Navigate to a URL
    keywordSel = 'input#txtKeyword' # Selector for the keyword search box
    # $(keywordSel).autocomplete source: data # Turn on auto complete search

    # Submit search term, when the user presses enter
    $(keywordSel).bind 'enter', () -> goToUrl('/map/'+$(keywordSel).val())
    $(keywordSel).keyup (e) -> if e.keyCode == 13 then $(this).trigger 'enter'

    # Live interactions
    window.addHeatToMap = (so) -> liveInteractions.addToMap(so, heatmapModule)
    window.clearMap = -> liveInteractions.clearMap(heatmapModule)
    window.togglePositive = ->
      liveInteractions.toggleHeatMap(heatmapModule.positiveHeatmap)
    window.toggleNegative = ->
      liveInteractions.toggleHeatMap(heatmapModule.negativeHeatmap)

  google.maps.event.addDomListener window, 'load', initialize
```

6.10.2 map/heatmap-module.coffee

```
positiveLocationData = []
negativeLocationData = []
neutralLocationData = []

makeLocationObject = (locationObj) ->
  new (google.maps.LatLng)(locationObj.lat, locationObj.lng)

# Make heatmap location data
sentimentResults.forEach (sentObj) ->
  if sentObj.sentiment > 0
    positiveLocationData.push({
      location: makeLocationObject(sentObj.location)
      weight: sentObj.sentiment * 10
    })
  else if sentObj.sentiment < 0
    negativeLocationData.push({
      location: makeLocationObject(sentObj.location)
      weight: Math.abs sentObj.sentiment * 10
    })
  else
    neutralLocationData.push({
      location: makeLocationObject(sentObj.location)
      weight: 5
    })

# Apply heat map to map
PositivePointArray = new (google.maps.MVCArray) (positiveLocationData)
positiveHeatmap =
  new (google.maps.visualization.HeatmapLayer) (data: PositivePointArray)

negativePointArray = new (google.maps.MVCArray) (negativeLocationData)
negativeHeatmap =
  new (google.maps.visualization.HeatmapLayer) (data: negativePointArray)

neutralPointArray = new (google.maps.MVCArray) (neutralLocationData)
neutralHeatmap =
  new (google.maps.visualization.HeatmapLayer) (data: neutralPointArray)

# Define gradients
positiveGradient = [
  'rgba(255,255,255,0)', 'rgba(20,180,240, 0.2)', 'rgba(20,185,220,0.4)'
  'rgba(20,190,210, 0.6)', 'rgba(20,195,200, 0.8)', 'rgb(20,210,190)'
  'rgb(20,215,180)', 'rgb(20,220,170)', 'rgb(20,225,160)', 'rgb(20,230,150)'
  'rgb(20,235,140)', 'rgb(20,240,130)', 'rgb(20,245,120)', 'rgb(20,246,100)'
  'rgb(20,247,90)', 'rgb(20,248,85)', 'rgb(20,249,80)', 'rgb(21,250,75)'
  'rgb(21,251,70)', 'rgb(21,252,65)', 'rgb(21,253,60)', 'rgb(21,254,55)'
  'rgb(20,255,50)', 'rgb(20,255,45)', 'rgb(20,255,40)', 'rgb(20,255,35)'
  'rgb(19,255,30)', 'rgb(19,255,28)', 'rgb(19,255,25)', 'rgb(19,255,23)'
  'rgb(18,255,20)', 'rgb(17,255,19)', 'rgb(16,255,18)', 'rgb(15,255,17)'
  'rgb(14,255,16)', 'rgb(13,255,15)', 'rgb(12,255,14)', 'rgb(11,255,13)'
  'rgb(10,255,10)', 'rgb(9,255,5)', 'rgb(8,255,0)', 'rgb(7,255,0)'
  'rgb(6,255,0)', 'rgb(5,255,0)', 'rgb(4,250,0)', 'rgb(3,245,0)'
  'rgb(2,240,0)', 'rgb(1,235,0)', 'rgb(0,230,0)', 'rgb(0,225,0)'
  'rgb(0,220,0)', 'rgb(0,215,0)', 'rgb(0,210,0)', 'rgb(0,205,0)'
  'rgb(0,200,0)', 'rgb(0,195,0)', 'rgb(0,190,0)', 'rgb(0,185,0)'
]

negativeGradient = [
  'rgba(255,255,255,0)', 'rgba(255,255,0,0.2)', 'rgba(255,255,0,0.4)'
  'rgba(255,245,0,0.6)', 'rgba(255,235,0,0.8)', 'rgb(255,225,0)',
  'rgb(255,215,0)', 'rgb(255,200,0)', 'rgb(255,180,0)', 'rgb(255,160,0)',
  'rgb(255,150,0)', 'rgb(255,145,0)', 'rgb(255,140,0)', 'rgb(255,135,0)',
  'rgb(255,130,0)', 'rgb(255,125,0)', 'rgb(255,120,0)', 'rgb(255,115,0)',
```



```

'rgb(255,110,0)', 'rgb(255,95,0)', 'rgb(255,90,0)', 'rgb(255,80,0)',
'rgb(255,70,0)', 'rgb(255,60,0)', 'rgb(255,50,0)', 'rgb(255,45,0)',
'rgb(255,40,0)', 'rgb(255,35,0)', 'rgb(255,30,0)', 'rgb(255,25,0)',
'rgb(255,20,0)', 'rgb(255,19,0)', 'rgb(255,18,0)', 'rgb(255,17,0)',
'rgb(255,16,0)', 'rgb(255,15,0)', 'rgb(255,14,0)', 'rgb(255,13,0)',
'rgb(255,12,0)', 'rgb(255,11,0)', 'rgb(255,10,0)', 'rgb(255,9,0)',
'rgb(255,6,0)', 'rgb(255,4,0)', 'rgb(255,2,0)', 'rgb(255,0,0)',
'rgb(240,0,0)', 'rgb(220,0,0)', 'rgb(210,0,0)', 'rgb(190,0,0)', 'rgb(170,0,0)'
]

neutralGradient = [
  'rgba(160,160,160,0)', 'rgba(150,150,150,0.2)', 'rgba(140,140,140,0.5)',
  'rgba(130,130,130,0.7)', 'rgba(120,120,120,0.9)', 'rgb(100,100,100)',
  'rgb(90,90,90)', 'rgb(80,80,80)', 'rgb(70,70,70)', 'rgb(60,60,60)',
  'rgb(50,50,50)', 'rgb(40,40,40)', 'rgb(35,35,35)', 'rgb(30,30,30)'
]

# Set gradients
positiveHeatmap.set('gradient', positiveGradient)
negativeHeatmap.set('gradient', negativeGradient)
neutralHeatmap.set('gradient', neutralGradient)

# Set opacity
positiveHeatmap.set('opacity', 0.7);
negativeHeatmap.set('opacity', 0.7);

module.exports.positiveHeatmap = positiveHeatmap
module.exports.negativeHeatmap = negativeHeatmap
module.exports.neutralHeatmap = neutralHeatmap

```

6.10.3 map/live-interactions-module.coffee

```
# Adds a sentiment object to the heatmap
addHeatToMap = (sentimentObject, heatmapModule) ->

  updateLayer = (heatmapLayer, newItem) ->
    if newItem.weight < 0 then newItem.weight = Math.abs(newItem.weight)
    newData = heatmapLayer.getData().j # Get old data
    newData.push(newItem) # Add new result to it
    heatmapLayer.setData(newData) # set the updated array

  location = new (google.maps.LatLng) (sentimentObject.location.lat,
    sentimentObject.location.lng)
  newItem =
    location: location
    weight: sentimentObject.sentiment * 10

  heatmapLayer =
    if sentimentObject.sentiment > 0 then heatmapModule.positiveHeatmap
    else if sentimentObject.sentiment < 0 then heatmapModule.negativeHeatmap
    else heatmapModule.neutralHeatmap

  updateLayer(heatmapLayer, newItem)

  $('span#numRes').text (Number($('span#numRes').text())+1) # Increment counter

  oldPositiveHeatData = []
  oldNegativeHeatData = []

  # Removes all heatmap layers
  clearMap = (heatmapModule) ->
    tempPos = oldPositiveHeatData
    tempNeg = oldNegativeHeatData

    oldPositiveHeatData = heatmapModule.positiveHeatmap.getData().j
    oldNegativeHeatData = heatmapModule.negativeHeatmap.getData().j

    heatmapModule.positiveHeatmap.setData(tempPos)
    heatmapModule.neutralHeatmap.setData([])
    heatmapModule.negativeHeatmap.setData(tempNeg)

  toggleHeatMap = (heatmap) ->
    heatmap.setMap if heatmap.getMap() then null else map

  module.exports.addToMap = addHeatToMap
  module.exports.clearMap = clearMap
  module.exports.toggleHeatMap = toggleHeatMap
```

6.10.4 Map/marker-cluster-module.coffee

```
removeWords = require 'remove-words'

class MarkerClusterSetup

  constructor: (map) -> @map = map

  # Creates the HTML for the info window displayed when a maker is clicked
  makeInfoWindowContent = (markerData) ->

    uniformWord = (word) -> (''+word).toLowerCase().replace /\W/g, ''

    # Make the coloured score string (e.g. '30% Positive')
    scoreString =
      if markerData.sentiment > 0
        "<b style='color: green;'>#{markerData.sentiment*100}% Positive</b>"
      else if markerData.sentiment < 0
        "<b style='color: darkred;'>#{markerData.sentiment*-100}% Negative</b>"
      else "<b style='color: grey;'>Neutral</b>"

    # Make clickable Tweet text
    clickWords = removeWords markerData.tweet # Array of keywords
    htmlTweet = ''
    aStyle = 'style="color: black; font-weight: bold;" ' # style for hyperlinks
    for word in markerData.tweet.split " "
      if uniformWord(word) in clickWords
        htmlTweet += "<a #{aStyle} href='/map/#{uniformWord word}'>#{word}</a> "
      else htmlTweet += "#{word} "

    # Put everything together to return
    "<div style='max-width: 25em'><p>#{htmlTweet}</p>#{scoreString}</div>"

  makeShape = () ->
    coords: [1, 1, 1, 20, 18, 20, 18, 1]
    type: 'poly'

  makeImage = (path) ->
    url: path
    size: new google.maps.Size(25, 25)
    origin: new google.maps.Point(0, 0)
    anchor: new google.maps.Point(12, 12)

  makeMarkers = (image, shape) ->
    infowindow = new google.maps.InfoWindow() # the pop-up thingy
    markers = [] # List of markers to be populated and returned

    # For each sentiment result, create a marker and push it to the array
    sentimentResults.forEach (sentObj) ->
      latLng = new google.maps.LatLng(sentObj.location.lat, sentObj.location.lng)
      marker = new google.maps.Marker({
        position: latLng
        map: @map
        icon: image
        shape: shape
        title : sentObj.tweet
      })
      # Action listener to show window when user presses marker
      google.maps.event.addListener marker, 'click', (evt) ->
        infowindow.setContent makeInfoWindowContent sentObj
        infowindow.open @map, this

    markers.push(marker)

  markers

  makeStyles = (path) ->
```

```

maxZoom: 100
gridSize: 40
styles: [
  {
    url: path
    height: 35
    width: 35
    anchor: [16, 0]
    textColor: 'transparent'
    textSize: 10
  }
  {
    url: path
    height: 45
    width: 45
    anchor: [24, 0]
    textColor: 'transparent'
    textSize: 11
  }
  {
    url: path
    height: 55
    width: 55
    anchor: [32, 0]
    textColor: 'transparent'
    textSize: 12
  }
]

start: ->
  pathToNothing = '/images/nothing.png'
  shape = makeShape()
  image = makeImage pathToNothing
  markers = makeMarkers image, shape
  styles = makeStyles pathToNothing

  new MarkerClusterer @map, markers, styles

# Export the stuff
module.exports = (map) -> (new MarkerClusterSetup map).start()

```

6.10.5 map/options-module.coffee

```
mapOptions =
  center: new (google.maps.LatLng) (51.5068, -0.1225)
  zoom: 3
  maxZoom: 20
  streetViewControl : false
  panControl: false
  mapTypeControlOptions:
    style: google.maps.MapTypeControlStyle.HORIZONTAL_BAR
    position: google.maps.ControlPosition.LEFT_TOP
    mapTypeIds: [
      'map_style'
      google.maps.MapTypeId.SATELLITE
    ]
  zoomControlOptions: {
    style: google.maps.ZoomControlStyle.LARGE
    position: google.maps.ControlPosition.LEFT_CENTER
  }

module.exports.mapOptions = mapOptions
```

6.10.6 map/socket-module.coffee

```
if searchTerm == ''
  socket = io.connect()

  socket.on 'tweet', (tweetObj) ->
    heatMapItem =
      sentiment: tweetObj.sentiment
      location:
        lat: tweetObj.location.location.lat
        lng: tweetObj.location.location.lng

    window.addHeatToMap (heatMapItem)
```

6.10.7 map/autocomplete-module.coffee

```
module.exports.autocomplete = $ ->
  $('input#txtKeyword').autocomplete
  source: data
  focusOpen: false
```

6.10.8 map/search-module.coffee

```
initiateSearch = (map) ->

  # Configure autocomplete and map search
  input = document.getElementById('txtLocation')
  autocomplete = new (google.maps.places.Autocomplete)(input)
  infowindow = new (google.maps.InfoWindow)

  # Prepare for putting markers on map
  marker = new (google.maps.Marker) (
    map: map
    anchorPoint: new (google.maps.Point)(0, -29)
  )
  google.maps.event.addListener autocomplete, 'place_changed', ->
    infowindow.close()
    marker.setVisible false
    place = autocomplete.getPlace()
    if !place.geometry
      window.alert 'Autocomplete\'s returned place contains no geometry'
      return

    # If the place has a geometry, then display marker on the map.
    if place.geometry.viewport
      map.fitBounds place.geometry.viewport
    else
      map.setCenter place.geometry.location
      map.setZoom 17 # Why 17? Because it looks good.
    marker.setIcon
      url: place.icon
      size: new (google.maps.Size)(71, 71)
      origin: new (google.maps.Point)(0, 0)
      anchor: new (google.maps.Point)(17, 34)
      scaledSize: new (google.maps.Size)(35, 35)
    marker.setPosition place.geometry.location
    marker.setVisible true
    address = ''
    if place.address_components
      address_parts = place.address_components
      address = [
        address_parts[1] and address_parts[1].short_name or ''
        address_parts[0] and address_parts[0].short_name or ''
        address_parts[2] and address_parts[2].short_name or ''
      ].join(' ')
    infowindow.setContent "<div><strong>#{place.name}</strong><br>" + address
    infowindow.open map, marker
    return

module.exports.initiatePlaceSearch = initiateSearch
```

6.10.9 map/theme-module.coffee

```
# Array of styles.
styles = [
  {
    'featureType': 'administrative.country'
    'elementType': 'geometry.stroke'
    'stylers': [ { 'color': '#DCE7EB' } ]
  }
  {
    'featureType': 'administrative.province'
    'elementType': 'geometry.stroke'
    'stylers': [ { 'color': '#DCE7EB' } ]
  }
]
```

```

    {
      'featureType': 'landscape'
      'elementType': 'geometry'
      'stylers': [ { 'visibility': 'off' } ]
    }
    {
      'featureType': 'poi'
      'elementType': 'all'
      'stylers': [ { 'visibility': 'off' } ]
    }
    {
      'featureType': 'road'
      'elementType': 'all'
      'stylers': [ { 'visibility': 'off' } ]
    }
    {
      'featureType': 'road'
      'elementType': 'labels'
      'stylers': [ { 'visibility': 'off' } ]
    }
    {
      'featureType': 'transit'
      'elementType': 'labels.icon'
      'stylers': [ { 'visibility': 'off' } ]
    }
    {
      'featureType': 'transit.line'
      'elementType': 'geometry'
      'stylers': [ { 'visibility': 'off' } ]
    }
    {
      'featureType': 'transit.line'
      'elementType': 'labels.text'
      'stylers': [ { 'visibility': 'off' } ]
    }
    {
      'featureType': 'transit.station.airport'
      'elementType': 'geometry'
      'stylers': [ { 'visibility': 'off' } ]
    }
    {
      'featureType': 'transit.station.airport'
      'elementType': 'labels'
      'stylers': [ { 'visibility': 'off' } ]
    }
    {
      'featureType': 'water'
      'elementType': 'geometry'
      'stylers': [ { 'color': '#83888B' } ]
    }
    {
      'featureType': 'water'
      'elementType': 'labels'
      'stylers': [ { 'visibility': 'off' } ]
    }
  ]

  module.exports.styledMap = new (google.maps.StyledMapType)(styles,
    name: 'Sentiment Map'
  )

```

6.10.10 globe/globe-main.coffee

```
ConfigureGlobe = require '../globe/configure-globe-module.coffee'
socketModule = require('../globe/socket-module.coffee')

configureGlobe = new ConfigureGlobe()

configureGlobe.go()

window.addSentiment = (so) -> configureGlobe.addNewPoint(so)

mainPage = 'globe'

require '../page-controls-module.coffee'
```

6.10.11 globe/socket-module.coffee

```
socket = io.connect();

socket.on 'tweet', (tweetObj) ->
  globeItem =
    sentiment: tweetObj.sentiment
    location:
      lat: tweetObj.location.location.lat
      lng: tweetObj.location.location.lng

  window.addSentiment(globeItem)

$('span#numRes').text(Number($('span#numRes').text()+1) # Increment counter
```

6.10.12 globe/configure-globe.coffee

```
class ConfigureGlobe

  constructor: -> @globe

  constants = {
    maxHeight: 4 # Maximum height of the bars on the globe
    neutralSentiment: 0.2 # The sentiment weighting to be assigned to neutral
    colors: [
      'rgb(240,40,40)' # Negative Color (red)
      'rgb(180,180,180)' # Neutral Color (gray)
      'rgb(40,240,40)' # Positive Color (green)
    ]
    containerId: 'container' # The ID of the HTML container to place the globe
    imageDir: '/images/' # The directory storing all globe-related images
  }

  makeBarHeight = (sentiment) ->
    if sentiment == 0 then sentiment = constants.neutralSentiment
    height = sentiment * constants.maxHeight
    if height >= 0 then height else Math.abs height # Always return positive

  makeBarColIndex = (sentiment) ->
    if sentiment > 0 then return 2
    else if sentiment < 0 then return 0
    else return 1
```



```

makeGlobeData = (sentimentResults) ->
  series = []
  for tweetObject in sentimentResults
    series.push tweetObject.location.lat
    series.push tweetObject.location.lng
    series.push makeBarHeight tweetObject.sentiment
    series.push makeBarColIndex tweetObject.sentiment
  [ [ 'Sentiment', series ] ];

createGlobe = () ->
  new (DAT.Globe)(document.getElementById(constants.containerId), {
    imgDir: constants.imageDir,
    colorFn: (label) -> new THREE.Color(constants.colors[label])
  })

addNewPoint: (sentimentObj) ->
  data = [
    sentimentObj.location.lat
    sentimentObj.location.lng
    makeBarHeight sentimentObj.sentiment
    makeBarColIndex sentimentObj.sentiment
  ]
  @globe.addData(data, {format: 'legend', name: 'newData'})
  @globe.createPoints()

go: () ->

  # Configure new globe
  @globe = createGlobe()

  # Make globe data array
  data = makeGlobeData sentimentResults

  # Add data to the globe
  for d in data then @globe.addData(d[1], {format: 'legend', name: d[0]})

  @globe.createPoints() # Create the geometry

  @globe.animate() # Begin animation

module.exports = ConfigureGlobe

```

6.10.13 now/now-main.coffee

\$ ->

```
# Include browserify modules
mapModule = require '../now/now-map-module.coffee'
barsModule = require '../now/now-bars-module.coffee'
countModule = require '../now/now-count-module.coffee'
topTwModule = require '../now/now-top-tweets-module.coffee'

# Initial render of each chart
mapModule.startDraw()
barsModule.generateBars()
countModule.startCountCharts()
topTwModule.init()

# Call relevant stuff when a new tweet arrives
newTweetArrived = (tweet) ->
  mapModule.addTweetToMap tweet
  barsModule.addToStats tweet

# Socket.io
socket = io.connect()
socket.on 'tweet', (tweetObj) ->
  if tweetObj.sentiment != 0
    newTweetArrived(tweetObj)
socket.on 'anyTweet', (tweetObj) ->
  if tweetObj.sentiment != 0
    countModule.newTweetArrived tweetObj
    if tweetObj.body.indexOf('http') == -1 and
      (tweetObj.sentiment > 0.6 or tweetObj.sentiment < -0.6)
      topTwModule.addTweet(tweetObj)

# Bit of page setup
$('input#txtSearch').characterCounter();
```

6.10.14 now/now-count-module.coffee

```
# Keep count of tweets
numPositiveTweets = 0
numNegativeTweets = 0
numTotal = 0

# Declare vars for objects
svg = null
chart = null

# Template for bullet data
bulletData = [{
  title: 'Sentiment'
  subtitle: 'Proportion of tweets'
  ranges: [0,1,2]
  measures: [1,2]
  markers: [1]
}]

drawBulletChart = () ->
  margin = top: 5, right: 40, bottom: 20, left: 120
  width = $('#bullet-container').width() - (margin.left) - (margin.right)
  height = 50 - (margin.top) - (margin.bottom)
  chart = d3.bullet().width(width).height(height)

  svg = d3.select('#bullet-chart')
    .selectAll('svg')
    .data(bulletData)
    .enter()
    .append('svg')
    .attr('class', 'bullet')
    .attr('width', width + margin.left + margin.right)
    .attr('height', height + margin.top + margin.bottom)
    .append('g')
    .attr('transform', 'translate(' + margin.left + ', ' + margin.top + ')')
    .call(chart)

  title = svg.append('g')
    .style('text-anchor', 'end')
    .attr('transform', 'translate(-6, ' + height / 2 + ')')
  title.append('text').attr('class', 'title').text(d) -> d.title
  title.append('text')
    .attr('class', 'subtitle')
    .attr('dy', '1em')
    .text(d) -> d.subtitle

addDataToBullet = () ->
  bulletData.ranges = [numTotal * 0.33, numTotal * 0.66, numTotal]
  bulletData.measures = [numPositiveTweets, numPositiveTweets+numNegativeTweets]
  bulletData.markers = [(numPositiveTweets + numNegativeTweets)/2]
  svg.datum(bulletData).call chart.duration(1000)

newTweetArrived = (tweet) ->
  numTotal += 1
  if tweet.sentiment > 0
    numPositiveTweets += 1
    $('#countPos').html(numPositiveTweets)
  else if tweet.sentiment < 0
    numNegativeTweets += 1
    $('#countNeg').html(numNegativeTweets)

  if (numTotal % 15) == 0
    addDataToBullet()
module.exports.startCountCharts = drawBulletChart
module.exports.newTweetArrived = newTweetArrived
```

6.10.15 now/now-bars-module.coffee

```
# Initialise Count Variables
numPositiveOfTweets = 0
numNegativeOfTweets = 0
totalPositiveScore = 0
totalNegativeScore = 0
totalNumTweets = 0

colChart = null

# Generate Initial Bar Chart
generateBars = () ->
  colChart = c3.generate(
    bindto: '#mini-bar-charts'
    data:
      columns: [ ['Positive', 0], ['Negative', 0] ]
      type: 'bar'
      colors:
        Positive: '#82FA58'
        Negative: '#F79F81'
      bar: width: ratio: 0.5
      axis:
        y:
          label: text: 'Weighted Proportions', position: 'outer-center'
          tick: format: -> return ''
  )

addToStats = (tweet) ->
# Update attributes and make calculations
s = tweet.sentiment # Get the sentiment
totalNumTweets += 1
if s > 0 # Positive Tweet!
  numPositiveOfTweets += 1
  totalPositiveScore += s
else if s < 0 # Negative Tweet!
  numNegativeOfTweets += 1
  totalNegativeScore += Math.abs s

# Update data visualizations
if s > 0 # Positive
  colChart.load columns: [['Positive', totalPositiveScore]]
else if s < 0 # Negative
  colChart.load columns: [['Negative', totalNegativeScore]]

module.exports.addToStats = addToStats
module.exports.generateBars = generateBars
```

6.10.16 now/now-map-modue.coffee

```
# Initialise geo config
geoChart = null

regionData = [
  ['Lat', 'Long', 'Sentiment', 'Size', {role: 'tooltip', p:{html:true}}]
  [51.2, -2.54, 0.1, 0.1, '']
]

geoOptions = {
  colorAxis: {colors: ['#DF0101', '#BDBDBD', '#04B404']}
  backgroundColor: '#2C2C2C'
  datalessRegionColor: '#D8D8D8'
  defaultColor: '#f5f5f5'
  showZoomOut: true
}

# Render region map
drawRegionsMap = ->
  data = google.visualization.arrayToDataTable(regionData)
  geoChart =
    new (google.visualization.GeoChart)(document.getElementById('geo-chart'))
  geoChart.draw data, geoOptions

# Add tweet to map
addTweetToMap = (tweet) ->
  regionData.push [tweet.location.location.lat, tweet.location.location.lng,
  tweet.sentiment, Math.abs(tweet.sentiment), tweet.body]
  if google.visualization?
    newMapData = google.visualization.arrayToDataTable(regionData)
    geoChart.draw newMapData, geoOptions
  if regionData.length > 5 && regionData.length < 10
    $('#geo-chart-loader').fadeOut('slow')
    $('#geo-chart').slideDown('slow')

startDraw = () ->
  google.charts.load 'current', 'packages': [ 'geochart' ]
  google.charts.setOnLoadCallback drawRegionsMap

module.exports.addTweetToMap = addTweetToMap
module.exports.startDraw = startDraw
```

6.10.17 now/now-top-tweets-module.coffee

```
initTopTweets = () ->

makeClickWords = (clickWords, body) ->
  clWord = (word) -> (''+word).toLowerCase().replace /\W/g, ''
  htmlTweet = ''
  aStyle = 'style="color: black; font-weight: bold;" ' # style for hyperlinks
  for word in body.split " "
    if clWord(word) in clickWords
      htmlTweet += "<a #{aStyle} href='/search/#{clWord word}'>#{word}</a> "
    else htmlTweet += "#{word} "
  htmlTweet

addTopTweet = (tweet) ->

  limit = 8

  tweetHtml = ""
  tweetHtml += "<div class='top-tweet' style='display: none;'>"
  tweetHtml += "#{makeClickWords(tweet.keywords, tweet.body)}"
  tweetHtml += "<span class='small-grey'>"
  tweetHtml += "<i class='tiny material-icons small-grey'>location_on</i>"
  tweetHtml += "#{tweet.location.place_name}</span>"
  tweetHtml += "</div>"

  if tweet.sentiment > 0
    $('#topPosLoader').slideUp('slow').remove()
    $('#topPositive').prepend(tweetHtml)
    $('#topPositive .top-tweet:first').slideDown('normal')
    .css('border-color', '#82FF79')
    if $('#topPositive .top-tweet').length >= $('#posLimit').val()
      $('#topPositive .top-tweet:last').slideUp('normal').remove()
  else if tweet.sentiment < 0
    $('#topNegLoader').slideUp('slow').remove()
    $('#topNegative').prepend(tweetHtml)
    $('#topNegative .top-tweet:first').slideDown('normal')
    .css('border-color', '#FF8675')
    if $('#topNegative .top-tweet').length >= $('#negLimit').val()
      $('#topNegative .top-tweet:last').slideUp('normal').remove()

$(document).ready -> $('select').material_select()

module.exports.init = initTopTweets
module.exports.addTweet = addTopTweet
```

6.10.18 visualisations/break-down.coffee

```
margin = { top: 40, right: 10, bottom: 10, left: 10 }
width = 960 - (margin.left) - (margin.right)
height = 500 - (margin.top) - (margin.bottom)

scaleColors = ["#a50026", "#d73027", "#f46d43", "#fdae61", "#fee08b", "#B4B4B4",
  "#d9ef8b", "#a6d96a", "#66bd63", "#1a9850", "#006837"]

color = d3.scale.linear()
  .domain([-1, -0.8, -0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.8, 1])
  .range(scaleColors)

treemap = d3.layout.treemap()
  .size([width, height])
  .sticky(true).value((d) -> d.size )

div = d3.select('div#tree')
  .append('div')
  .style('position', 'relative')
  .style('width', width + margin.left + margin.right + 'px')
  .style('height', height + margin.top + margin.bottom + 'px')
  .style('left', margin.left + 'px')
  .style('top', margin.top + 'px')

position = ->
  @style('left', (d) -> d.x + 'px')
  .style('top', (d) -> d.y + 'px')
  .style('width', (d) -> Math.max(0, d.dx - 1) + 'px' )
  .style 'height', (d) -> Math.max(0, d.dy - 1) + 'px'

root = data
node = div.datum(root)
  .selectAll('.tree-node')
  .data(treemap.nodes)
  .enter()
  .append('div')
  .attr('class', (d) -> 'tree-node ' + if d.topic? then 'tooltipped')
  .call(position)
  .attr('data-position', 'top')
  .attr('data-tooltip', (d) -> if d.topic != '' then d.topic else '[no topic]')
  .style('background', (d) -> if d.children then color(d.name) else null)
  .text((d) -> if d.children then null else d.name )

d3.selectAll('input').on 'change', ->
  value = if @value == 'count' then (-> 1) else ((d) -> d.size)
  node.data(treemap.value(value).nodes)
  .transition()
  .duration(1500)
  .call position

$(document).ready ->
  $('#tooltipped').tooltip delay: 50

$('#txtKeyword').keyup (e) -> if e.keyCode == 13
  showLoader()
  window.location = '/break-down/'+$('#txtKeyword').val()
```

6.10.19 visualisations/comparer.coffee

```
window.drawDonuts = (comparisonData) ->
  for result, index in comparisonData
    chart = c3.generate(
      bindto: '#chart-'+index
      data:
        columns: [
          ['Positive', result.pieChart.positive ]
          # ['Neutral', result.pieChart.neutral ]
          ['Negative', result.pieChart.negative ]
        ]
      type: 'donut'
      colors: {
        Positive: '#01DF01',
        Neutral: '#848484',
        Negative: '#DF0101'
      }
      donut: title: result.searchTerm)

$('#get-results').click () ->
  url = '/comparer/'
  if $('#brand-1').val() != '' then url += $('#brand-1').val()
  if $('#brand-2').val() != '' then url += ','+$('#brand-2').val()
  if $('#brand-3').val() != '' then url += ','+$('#brand-3').val()
  if $('#brand-4').val() != '' then url += ','+$('#brand-4').val()
  showLoader()
  window.location = url

$('#a#add-new').click () ->
  $('.input-field').slideDown('fast')
  $(this).fadeOut('fast')

$('#brand-2').keyup () ->
  if $(this).val() != ''
    $('.input-field').slideDown('fast')
    $('#a#add-new').fadeOut('fast')
```

6.10.20 visualisations/comparison.coffee

```
# Converts the raw Sentiment data into a format suitable for the scatter chart
generateScatterData = (rawResults) ->
  chartData = []
  chartData.push(['Sentiment', 'Dictionary-Based SA', 'NLU SA', 'Human SA'])
  chartData.push([0, null, null, 0])
  rawResults.forEach (sentimentObject) ->
    tweetLen = sentimentObject.tweet.length
    chartData.push([tweetLen, null, sentimentObject.dictionary_sentiment, null])
    chartData.push([tweetLen, sentimentObject.nlu_sentiment, null, null])
    if (sentimentObject.human_sentiment)
      chartData.push([tweetLen, null, null, sentimentObject.human_sentiment])
  chartData

# Converts raw sentiment data into a format suitable for the bar chart
generateBarData = (rawResults) ->
  chartData = []
  chartData.push(['Tweet', 'Dictionary-Based SA', 'NLU SA', 'Human SA'])
  rawResults.forEach (so) ->
    chartData.push([
      so.tweet
      so.nlu_sentiment
      so.dictionary_sentiment
      if so.human_sentiment? then so.human_sentiment else 0
    ])
```



```

    ])
    chartData

# Converts raw sentiment data into a format suitable for the column chart
generateSummaryData = (rawResults) ->
  chartData = []
  chartData.push ['', 'Dictionary', 'NLU', 'Human']
  dictionaryTotal = 0
  nluTotal = 0
  humanTotal = 0
  rawResults.forEach (sentimentObject) ->
    dictionaryTotal += sentimentObject.dictionary_sentiment
    nluTotal += sentimentObject.nlu_sentiment
    humanTotal += sentimentObject.human_sentiment
  chartData.push([
    'Sentiment Analysis Results'
    dictionaryTotal/rawResults.length
    nluTotal/rawResults.length
    humanTotal/rawResults.length
  ])
  chartData

# Finds range of sentiment
getSummaryRange = (chartData) ->
  chartData[1].splice 0,1
  for d, i in chartData[1] then chartData[1][i] = Math.abs(d)
  Math.max.apply(Math, chartData[1])*2

# Defines the chart data, options and calls draw method for both charts
drawChart = ->
  scatterData =
    google.visualization.arrayToDataTable generateScatterData sentimentResults
  barData =
    google.visualization.arrayToDataTable generateBarData sentimentResults
  summaryData =
    google.visualization.arrayToDataTable generateSummaryData sentimentResults

# Define chart options for all charts
scatterOptions =
  title: 'Comparison of Different Sentiment Analysis Methods'
  width: 900
  height: 500
  backgroundColor: '#fff'
  hAxis:
    title: 'String Length'
    minValue: 0
    maxValue: 100
  vAxis:
    title: 'Sentiment'
    minValue: -1
    maxValue: 1
  legend: position: 'right'
  colors: ['#6ec9ee', '#3D7CDB', '#0B326C']
  dataOpacity: 1

barOptions =
  chart:
    title: 'Comparison of different sentiment analysis methods '
    subtitle: 'Hover over bar for more details of Tweet'
    width: 1000
    height: 1500
    chartArea:
      width: '50%'
      height: '100%'
    backgroundColor: '#fff'
    bars: 'horizontal'
    colors: ['#6ec9ee', '#3D7CDB', '#0B326C']
    axes: x: 0:
      side: 'top'

```

```

    label: 'Sentiment'

summaryOptions =
  chart:
    title: 'Summary of Sentiment Analysis Results'
    subtitle: 'Comparing Dictionary and NLU based results'
    bars: 'vertical'
    vAxis:
      format: 'decimal'
      viewWindowMode: 'explicit'
      viewWindow:
        min: - getSummaryRange generateSummaryData sentimentResults
        max: getSummaryRange generateSummaryData sentimentResults
    height: 400
    colors: ['#6ec9ee', '#3D7CDB', '#0B326C']

# Create all google charts with chart options
chart = new google.charts.Bar document.getElementById 'summary_bar_ch'
chart.draw summaryData, google.charts.Bar.convertOptions summaryOptions

scatterChart =
  new google.visualization.ScatterChart document.getElementById 'scatter_ch'
scatterChart.draw scatterData, scatterOptions

barChart = new google.visualization.BarChart document.getElementById 'bar_ch'
barChart.draw barData, barOptions

# Create a HTML table for the raw results
drawTableChart = () ->
  data = generateBarData sentimentResults
  myTable = ''
  myTable += '<thead><tr>'
  myTable += '<th style="width: 30em">Tweet</th>'
  myTable += '<th>'+item+'</th>' for item in data[0].splice(1,3)
  myTable += '</tr></thead>'
  myTable += '<tbody>'
  i = 1
  while i < data.length
    myTable += '<tr>'
    myTable += '<td>'+cell+'</td>' for cell in data[i]
    myTable += '</tr>'
    i++

  myTable += '</tbody>'

  document.getElementById('table_ch').innerHTML = drawTableChart()

# Load the google visualisations packages and call the draw method
google.load 'visualization', '1.1', packages: ['table', 'corechart', 'bar']
google.setOnLoadCallback drawChart

# Update the search button when the textfield changes
btnSearch = document.getElementById('btnCalculate')
txtKeyword = document.getElementById('txtKeyword')
txtKeyword.onchange = txtKeyword.onkeyup = ->
  keyWord = encodeURIComponent(txtKeyword.value)
  btnSearch.setAttribute('href', '/sa-comparison/'+keyWord)

```

6.10.21 visualisations/comparison-for-search.coffee

```
requestDbData = (tweetBody) ->

makePieData = (tweetArr) ->
  res = {searchTerm: '', pieChart: {}}
  pieChart = {positive: 0, neutral: 0, negative: 0}
  for tweet in tweetArr
    if tweet.sentiment > 0 then pieChart.positive += 1
    else if tweet.sentiment < 0 then pieChart.negative += 1
    else pieChart.neutral += 1
  res.pieChart = pieChart
  res

# Called after results are returned, initiates the rendering process
renderResults = (dbRes) ->
  window.drawDonuts([makePieData(results), makePieData(dbRes)])

# Make the actual request
$.post('/api/db', {}, (results) -> renderResults results)

$(document).ready ->
  requestDbData tweetBody
```

6.10.22 visualisations/entity-extraction.coffee

```
units = 'Widgets'
margin = top: 10, right: 10, bottom: 10, left: 10
width = 920 - (margin.left) - (margin.right)
height = 540 - (margin.top) - (margin.bottom)
formatNumber = d3.format(',.0f')

format = (d) -> formatNumber(d) + ' ' + units

color = d3.scale.category20()

# append the svg canvas to the page
svg = d3.select('#chart')
  .append('svg')
  .attr('width', width + margin.left + margin.right)
  .attr('height', height + margin.top + margin.bottom)
  .append('g')
  .attr('transform', 'translate(' + margin.left + ', ' + margin.top + ')')

# Set the sankey diagram properties
sankey = d3.sankey().nodeWidth(36).nodePadding(10).size([width, height])
path = sankey.link()

# load the data
graph = sankeyData
nodeMap = {}

# the function for moving the nodes
dragmove = (d) ->
  d3.select(this)
    .attr('transform', 'translate(' + (
      d.x = Math.max(0, Math.min(width - (d.dx), d3.event.x))
    ) + ', ' + (
      d.y = Math.max(0, Math.min(height - (d.dy), d3.event.y))
    ) + ')')
  sankey.relayout()
  link.attr('d', path)

graph.nodes.forEach (x) -> nodeMap[x.name] = x
```

```

graph.links = graph.links.map((x) ->
  {
    source: nodeMap[x.source]
    target: nodeMap[x.target]
    value: x.value
  }
)

sankey.nodes(graph.nodes).links(graph.links).layout 32

# add in the links
link = svg.append('g')
  .selectAll('.sankey-link')
  .data(graph.links)
  .enter()
  .append('path')
  .attr('class', 'sankey-link')
  .attr('d', path)
  .style('stroke-width', (d) -> Math.max 1, d.dy)
  .sort((a, b) -> b.dy - (a.dy) )

# add the link titles
link.append('title').text (d) ->
  d.source.name + ' → ' + d.target.name + '\n' + format(d.value)

# add in the nodes
node = svg.append('g')
  .selectAll('.sankey-node')
  .data(graph.nodes)
  .enter()
  .append('g')
  .attr('class', 'sankey-node')
  .attr('transform', (d) ->'translate(' + d.x + ',' + d.y + ')')
  .call(d3.behavior.drag().origin((d) -> d)
    .on('dragstart', -> @parentNode.appendChild this)
    .on('drag', dragmove))

# add the rectangles for the nodes
node.append('rect').attr('height', (d) -> d.dy)
  .attr('width', sankey.nodeWidth())
  .style('fill', (d) -> d.color = color(d.name.replace(RegExp('.*'), '')))
  .style('stroke', (d) -> d3.rgb(d.color).darker 2)
  .append('title').text (d) -> d.name + '\n' + format(d.value)

# add in the title for the nodes
node.append('text')
  .attr('x', -6)
  .attr('y', (d) -> d.dy / 2)
  .attr('dy', '.35em')
  .attr('text-anchor', 'end')
  .attr('transform', null).text((d) -> d.name)
  .filter((d) -> d.x < width / 2)
  .attr('x', 6 + sankey.nodeWidth())
  .attr('text-anchor', 'start')

# Submit search term when enter is pressed
$('#txtKeyword').keyup (e) -> if e.keyCode == 13
  showLoader()
  window.location = '/entity-extraction/' + $('#txtKeyword').val()

```

6.10.23 visualisations/entity-summary.coffee

```
requestEntityData = (tweetBody) ->

# Generates the HTML for each progress bar with label, value and tooltip
makeHtmlProgress = (label, img, num) ->
  html = ""
  html += "<a href='/search/#{label}'>"
  html += "<div class='chip sml-margin tooltiped' data-tooltip='#{num}
occurrences'>"
  if img? then if img != '' then html += "<img src='#{img}' />"
  html += "#{label}"
  html += "</div>"
  html += "</a>"
  html

# Called after results are returned, initiates the rendering process
renderResults = (results) ->
  i = 0
  for category in results
    i += 1
    $('#entityResults'+i).append(
      "<h5 class='flow-text'>#{category.name}</h5>"
    )
    for item in category.items
      img = item.additional_information.image
      $('#entityResults'+i).append(makeHtmlProgress(item.normalized_text, img,
item.matches.length))
      # Show containers now they have data in, and hide the loader
      $('#entityLoader').fadeOut('fast')
      j = 1
      while j <= 8 then $('#entityResults'+j).slideDown('slow'); j++
      $('img').error -> $(this).hide() # Hide 404 not found images
      $('.tooltiped').tooltip({delay: 50}) # Initialise the tooltip

# Hide empty containers to start with
j = 1
while j <= 8 then $('#entityResults'+j).hide(); j++

# Make the actual request
$.post('/api/entity', {text:tweetBody}, (results) -> renderResults results)

$(document).ready ->
  requestEntityData tweetBody
```

6.10.24 visualisations/gauge-module.coffee

```
sentiment = (average * 100) / 2 + 50

chart = c3.generate(
  bindto: '#gaugeChart'
  data:
    columns: [ ['sentiment', sentiment] ]
    type: 'gauge'
  gauge: {}
  color:
    pattern: ["#C80000", "#EB3443", "#F85353", "#FF5A80", "#EF7B96", "#B1B1B1",
      "#8CD087", "#7CE974", "#62EC59", "#42F735", "#4AF43E"]
    threshold: values: [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
  size: height: 180)
```

6.10.25 visualisations/hexagons-module.coffee

```
# Hexagon color scale
scaleColors = ["#C80000", "#EB3443", "#F85353", "#FF5A80", "#EF7B96", "#B1B1B1",
  "#8CD087", "#7CE974", "#62EC59", "#42F735", "#4AF43E"]
fillScale = d3.scale.linear()
.domain([-1, -0.8, -0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.8, 1])
.range(scaleColors)

if homePage? and homePage
  scaleColors = ['rgb(15, 160, 255)', 'rgb(60, 230, 255)', 'rgb(60, 255, 181)'
    'rgb(70, 255, 99)', 'rgb(174, 255, 99)', 'rgb(217, 255, 99)'
    'rgb(251, 255, 99)', 'rgb(255, 230, 99)', 'rgb(255, 182, 99)'
    'rgb(255, 142, 99)', 'rgb(250, 125, 100)', 'rgb(255, 117, 99)'
    'rgb(255, 99, 100)', 'rgb(255, 55, 55)']
  ]
  scaleColors.reverse()
  fillScale = d3.scale.linear()
  .domain([-1, -0.8, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3,
    0.4, 0.5, 0.6, 0.8, 1])
  .range(scaleColors)

# Initialise tooltip
tip = d3.tip().attr('class', 'd3-tip').html((d, i) ->
  s = results[i].sentiment
  col = if s > 0 then 'green' else if s < 0 then 'darkred' else 'grey'
  html = '<b>Sentiment:</b>'
  html += ' <span style=\'color:' + col + '\>' + s + '</span> <br>'
  html += '<span>' + results[i].body + '</span>'
  html
)

#svg sizes and margins
margin = top: 30, right: 20, bottom: 20, left: 50
if homePage? and homePage then margin = top: 0, right: 0, bottom: 0, left: 0

width = $(window).width() - margin.left - margin.right
height = ($(window).height() - margin.top - margin.bottom - 80)/2
if hexPage? then if hexPage then height = height and width = width -= 10
else if homePage? then if homePage then height = height * 0.8 and width += 20

MapColumns = Math.round(Math.sqrt(results.length*1.33))
MapRows = Math.round(Math.sqrt(results.length*0.66))

#The maximum radius the hexagons can have to still fit the screen
hexRadius = d3.min([
  width / ((MapColumns + 0.5) * Math.sqrt(3))
  height / ((MapRows + 1 / 3) * 1.5)
])

#Set the new height and width of the SVG
width = MapColumns * hexRadius * Math.sqrt(3)
height = MapRows * 1.5 * hexRadius + 0.5 * hexRadius

hexbin = d3.hexbin().radius(hexRadius) #Set the hexagon radius

#Calculate the center positions of each hexagon
points = []
i = 0
while i < MapRows
  j = 0
  while j < MapColumns
    points.push [hexRadius * j * 1.75, hexRadius * i * 1.5]
    j++
  i++

#Create SVG element
```

```

svg = d3.select('#chart')
.append('svg')
.attr('width', width + margin.left + margin.right)
.attr('height', height + margin.top + margin.bottom)
.append('g')
.attr('transform', 'translate(' + margin.left + ',' + margin.top + ')')

svg.call(tip) # call tool tip

#Start drawing the hexagons
svg.append('g')
.selectAll('.hexagon')
.data(hexbin(points))
.enter()
.append('path')
.attr('class', 'hexagon')
.attr('d', (d) -> 'M' + d.x + ',' + d.y + hexbin.hexagon())
.attr('stroke', (d, i) -> '#fff')
.attr('stroke-width', '1px')
.style('fill', (d, i) -> fillScale(results[i].sentiment))
.on 'mouseover', (d, i) ->
  tip.show(d, i)
  el = d3.select(this).transition().duration(10).style('fill-opacity', 0.3)
.on 'mouseout', (d, i) ->
  tip.hide(d, i)
  el = d3.select(this).transition().duration(1000).style('fill-opacity', 1)

window.updateHexData = (newData) ->
  randomIndex = Math.floor(Math.random() * points.length)
  results[randomIndex] = newData
  svg.selectAll('.hexagon')
  .data(hexbin(points))
  .style('fill', (d, i) -> fillScale(results[i].sentiment))
  .enter()

# Socket.io
if io?
  socket = io.connect();
  socket.on 'anyTweet', (tweetObj) ->
    if tweetObj.sentiment != 0
      tweet = sentiment: tweetObj.sentiment, body: tweetObj.body
      window.updateHexData(tweet)

```

6.10.26 visualisations/homepage.coffee

```
homePage = true # Show different hexagons for the homepage

# Submit search field when enter is pressed
$('#txtKeyword').keyup (e) ->
  if e.keyCode == 13
    showLoader()
    window.location = '/search/' + $('#txtKeyword').val()
  return

$(document).ready ->

# Set the size of the chart to fit the window
$('#part-1').css 'height', $(window).height()
$('#chart').find('svg').attr
  'height': $(window).height() + 20
  'width': $(window).width()

# Pull down the blocks on the front page
$('.home-row').animate { 'margin-top': $(window).height() / 5 },
  duration: 600
  step: (now) -> $(this).attr 'margin-top', now

# Scroll to positions
$('#a.scroll-down#scroll-1').click ->
  $('html, body').animate { scrollTop: $('#part-2').offset().top }, 850

# Parallax scroll
parallaxScroll = ->
  scrolledY = $(window).scrollTop()
  $('#part-1').css 'margin-top', scrolledY * 0.5 + 'px'
  $('#scroll-1').css 'bottom', scrolledY * 0.3 + 10 + 'px'
  $('#hex-details').css 'bottom', scrolledY * 0.3 + 10 + 'px'

$(window).bind 'scroll', (e) -> parallaxScroll()
```

6.10.27

6.10.28 visualisations/radar-module.coffee

```
renderChart = (toneResults) ->

  $('#radarLoader').slideUp('slow')

  scaleColors = ['rgb(15, 160, 255)', 'rgb(60, 230, 255)', 'rgb(60, 255, 181)'
    'rgb(70, 255, 99)', 'rgb(174, 255, 99)', 'rgb(217, 255, 99)'
    'rgb(251, 255, 99)', 'rgb(255, 230, 99)', 'rgb(255, 182, 99)'
    'rgb(255, 142, 99)', 'rgb(250, 125, 100)', 'rgb(255, 117, 99)'
    'rgb(255, 99, 100)', 'rgb(255, 55, 55)', 'rgb(240,240,240)']
  ].reverse()
  color = d3.scale.linear()
  .domain([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,18])
  .range(scaleColors)

  width = 600
  height = 500
  radius = Math.min(width, height) / 2 - 10
  formatNumber = d3.format(',d')
  x = d3.scale.linear().range([0,2 * Math.PI])
  y = d3.scale.sqrt().range([0, radius])
  partition = d3.layout.partition().value((d) -> d.size)
  arc = d3.svg.arc().startAngle((d) -> Math.max 0, Math.min(2 * Math.PI, x(d.x)))
    .endAngle((d) ->Math.max 0, Math.min(2 * Math.PI, x(d.x + d.dx)))
    .innerRadius((d) -> Math.max 0, y(d.y))
    .outerRadius((d) -> Math.max 0, y(d.y + d.dy))

  svg = d3.select('#tone-radar')
    .append('svg')
    .attr('width', width)
    .attr('height', height)
    .append('g')
    .attr('transform', 'translate(' + width / 2 + ', ' + height / 2 + ')')

  click = (d) ->
    svg.transition().duration(750).tween('scale', ->
      xd = d3.interpolate(x.domain(), [ d.x, d.x + d.dx ])
      yd = d3.interpolate(y.domain(), [ d.y, 1 ])
      yr = d3.interpolate(y.range(), [ (if d.y then 20 else 0), radius ])
      (t) ->
        x.domain xd(t)
        y.domain(yd(t)).range yr(t)
    ).selectAll('path').attrTween 'd', (d) -> -> arc d

  root = {name: 'Tones', children: []}
  for category, i in toneResults.document_tone.tone_categories
    root.children[i] = name: category.category_name, children: []
    for tone, j in category.tones
      root.children[i].children[j] = {name: tone.tone_name, size: tone.score}

  svg.selectAll('path')
    .data(partition.nodes(root))
    .enter()
    .append('path')
    .attr('d', arc)
    .attr('class', 'tt')
    .attr('data-tooltip', (d) -> d.name)
    .style('fill', (d, i) -> color i)
    .on('click', click)
    .append('title')
    .text (d) -> d.name + '\n' + formatNumber(d.value)

  d3.select(self.frameElement).style 'height', height + 'px'
  $('tt').tooltip({delay: 50}) # Initialise the tooltip

module.exports = renderChart
```

6.10.29 visualisations/real-time-dash.coffee

```
# Color Scale
scaleColors = ["#a50026", "#c11940", "#d73027", "#dc4139", "#f04539", "#f5504d",
               "#fd6060", "#f87575", "#dd8e8e", "#b4b4b4", "#b4b4b4", "#bcf08c",
               "#d9ef8b", "#92e16e", "#a6d96a", "#71c96e", "#66bd63", "#21b04c",
               "#1a9850", "#027d35", "#006837"]

fillScale = d3.scale.linear()
  .domain([-1,-0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1,
          0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
  .range(scaleColors)

# Global Variables
chart = 0
chartData = 0
rawData = []

# Generate the structure for the initial data object
makeData = -> [{key: "Sentiment Data", values: []}]

# Generate NVd3 Graph with options
nv.addGraph ->
  chart = nv.models.scatterChart()
    .showDistX(true)
    .showDistY(true)
    .transitionDuration(350)
    .showLegend(false)
    .tooltipContent (key, y, e, graph) -> "<b style='max-width: 100ch'>
#{graph.point.label} </b> "

# Chart axis
chart.xAxis
  .axisLabel('Time (HH:MM)')
  .tickFormat (d) -> d3.time.format('%I:%M') new Date(d)
chart.yAxis
  .axisLabel('Sentiment (+/-)')
  .tickFormat (d) -> Math.round(d/10)*10 + '%'

# Specify data and render
rawData = makeData()
chartData = d3.select('#real-time-scatter svg').datum(rawData)
chartData.call chart
nv.utils.windowResize chart.update
chart

# Dynamically add a new point to the graph
window.addPoint = (time, sentiment, label) ->
  axisSent = sentiment + (Math.round((Math.random()/10)-0.05) *10000)/10000)
  size = Math.round(label.length/10) + (Math.abs(sentiment) * 10)
  rawData[0].values.push {
    x: time, y: axisSent * 100, size: size, label: label, color:
  fillScale(sentiment)
  }
  chartData.datum(rawData).transition().duration(500).call(chart)
  nv.utils.windowResize(chart.update)

$ ->
$( '#real-time-scatter' ).hide()
setTimeout(->
  $( '#scatter-loader' ).fadeOut('slow')
, 1500)

# Socket.io
if io?
  socket = io.connect();
  socket.on 'anyTweet', (tweetObj) ->
    if tweetObj.sentiment != 0 && tweetObj.body.indexOf('http') == -1
      window.addPoint(new Date().getTime(), tweetObj.sentiment, tweetObj.body)
```

6.10.30 visualisations/regions-main.coffee

```
pageControls = require '../page-controls-module.coffee'
pageControls.setMainPage 'region-map'

# Takes the raw CSV string and returns a nice list of JSON region objects
convertCsvToJson = (csvRegions) ->
  regions = []
  for r in csvRegions.splice 1, csvRegions.length # For each line in CSV file
    r = r.match(/(".*?"|^[^",\s]+) (?=\s*,|\s*$)/g) # Break into array
    for e, i in r then r[i] = r[i].replace(/[""]+/g, '').trim() # Neaten
    regions.push { # Create a JSON object for region, and push to results
      country: r[0], alpha2_code: r[1], alpha3_code: r[2],
      numeric_code: r[3], latitude: Number(r[4]), longitude: Number(r[5])
    }
  regions # Done, return regions

# Convert regions to JSON
regions = convertCsvToJson csvRegions

# Make data array
data = []
for r in regions then data.push value: r.alpha2_code, label: r.country

$(document).ready ->
  locSel = '#txtLocation'
  $(locSel).autocomplete(
    source: data
    focusOpen: false
    callback: (value) -> window.location = ('/region-map/location/'+value)
  )
  $(locSel).bind 'enter', () ->
    val = $(locSel).val()
    if val.length == 2 then window.location = ('/region-map/location/'+val)
    else alert("Invalid Country Code - select an option from the dropdown menu")
  $(locSel).keyup (e) -> if e.keyCode == 13 then $(this).trigger 'enter'

drawRegionsMap = ->

  data = google.visualization.arrayToDataTable(sentimentResults)
  options = {
    colorAxis: {colors: ['#DF0101', '#BDBDBD', '#04B404']}
    backgroundColor: '#2C2C2C'
    datalessRegionColor: '#D8D8D8'
    defaultColor: '#f5f5f5'
    showZoomOut: true
  }

  if searchRegion != '' then options.region = searchRegion

  chart = new
  (google.visualization.GeoChart)(document.getElementById('regions_div'))
  chart.draw data, options

google.charts.load 'current', 'packages': [ 'geochart' ]
google.charts.setOnLoadCallback drawRegionsMap
```

6.10.32

6.10.33 visualisations/render-tone-bars.coffee

```
# Generates the HTML for each progress bar with label, value and tooltip
makeHtmlProgress = (label, value) ->
  percent = Math.round(value*100)
  html = ""
  html += "<label>#{label}</label>"
  html += "<div class='progress horizontal-bar tooltiped' data-position='right'
data-tooltip='#{percent}%>"
  html += "<div class='determinate' style='width: #{percent}%>"
  html += "</div></div>"
  html

# Called after results are returned, initiates the rendering process
renderResults = (results) ->
  for tCat, i in results.document_tone.tone_categories
    i += 1
    $('#toneResults'+i).append(
      "<h5 class='flow-text font-size1'>#{tCat.category_name}</h5>"
    )
    for tone in tCat.tones
      $('#toneResults'+i).append(makeHtmlProgress(tone.tone_name, tone.score))
  # Show containers now they have data in, and hide the loader
  $('#toneLoader').fadeOut('fast')
  j = 1
  while j <= 3 then $('#toneResults'+j).slideDown('slow'); j++
  $('.tooltiped').tooltip({delay: 50}) # Initialise the tooltip

module.exports = renderResults
```

6.10.34 visualisations/search-summary-main.coffee

```
hexagonsModule = require '../visualisations/hexagons-module.coffee'
gaugeModule = require '../visualisations/gauge-module.coffee'
pageControls = require '../page-controls-module.coffee'
pageControls.setMainPage 'search'
```

6.10.36 visualisations/scatter-words-main.coffee

```
d3 = require 'd3/d3.min.js'
tipsy = require 'tipsy/index.js'

mainPage = 'word-scatter-plot'
pageControls = require '../page-controls-module.coffee'
pageControls.setMainPage mainPage

# set the stage
margin = {t: 30, r: 20, b: 20, l: 40}
w = 600 - (margin.l) - (margin.r)
h = 500 - (margin.t) - (margin.b)

biggestFreq = 10
for e, i in wordData
  if e.freq > biggestFreq then biggestFreq = e.freq
  ran = Math.round(Math.random()*10)/100
  wordData[i].sentiment = wordData[i].sentiment + ran

x = d3.scale.linear().range([0, w/4, w/2, (w/4)*3, w])
y = d3.scale.linear().range([h - 60, 0])

scaleColors = ["#a50026", "#d73027", "#fdae61", "#B4B4B4",
               "#a6d96a", "#1a9850", "#006837"]

color = d3.scale.linear()
  .domain([-0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6])
  .range(scaleColors)

svg = d3.select('#scatter-words')
  .append('svg')
  .attr('width', w + margin.l + margin.r)
  .attr('height', h + margin.t + margin.b)

# set axes, as well as details on their ticks
xAxis = d3.svg.axis().scale(x)
  .ticks(20).tickSubdivide(true).tickSize(6, 3, 0).orient('bottom')
yAxis = d3.svg.axis().scale(y)
  .ticks(20).tickSubdivide(true).tickSize(6, 3, 0).orient('left')

# group that will contain all of the plots
groups = svg.append('g')
  .attr('transform', 'translate(' + margin.l + ', ' + margin.t + ')')

# array of the sentiments, used for the legend
sentiments = ['Positive', 'Neutral', 'Negative' ]

x0 = Math.max(-d3.min(wordData, (d) -> d.freq
), d3.max(wordData, (d) -> d.freq
)
)
x.domain [0, 10, 20, 50, biggestFreq]
y.domain [-0.6, 0.6]

# style the circles, set their locations based on data
circles = groups.selectAll('circle')
  .data(wordData)
  .enter()
  .append('circle')
  .attr('class', 'circles')
  .attr(
    cx: (d) -> x + d.freq
    cy: (d) -> y + d.sentiment
    r: 8
    id: (d) -> d.text
  )
  .style('fill', (d) -> color d.sentiment )
  .style('opacity', '0.6')
```

```

# what to do when we mouse over a bubble
mouseOn = ->
circle = d3.select(this)
# transition to increase size/opacity of bubble
circle.transition()
  .duration(800)
  .style('opacity', 1)
  .attr('r', 16).ease 'elastic'

# append lines to bubbles that will be used to show the precise data points.
# translate their location based on margins
svg.append('g')
  .attr('class', 'guide')
  .append('line')
  .attr('x1', circle.attr('cx'))
  .attr('x2', circle.attr('cx'))
  .attr('y1', +circle.attr('cy') + 26)
  .attr('y2', h - (margin.t) - (margin.b))
  .attr('transform', 'translate(40,20)')
  .style('stroke', circle.style('fill'))
  .transition()
  .delay(200)
  .duration(400)
  .styleTween 'opacity', -> d3.interpolate 0, .5

svg.append('g')
  .attr('class', 'guide')
  .append('line')
  .attr('x1', +circle.attr('cx') - 16)
  .attr('x2', 0)
  .attr('y1', circle.attr('cy'))
  .attr('y2', circle.attr('cy'))
  .attr('transform', 'translate(40,30)')
  .style('stroke', circle.style('fill'))
  .transition().delay(200).duration(400)
  .styleTween 'opacity', -> d3.interpolate 0, .5

# function to move mouseover item to front of SVG stage, in case
# another bubble overlaps it
d3.selection::moveToFront = ->
  @each ->
    @parentNode.appendChild this

circle.moveToFront()

# what happens when we leave a bubble?
mouseOff = ->
circle = d3.select(this)

# go back to original size and opacity
circle.transition()
  .duration(800)
  .style('opacity', .4)
  .attr('r', 8)
  .ease 'elastic'

# fade out guide lines, then remove them
d3.selectAll('.guide').transition().duration(100).styleTween('opacity', ->
  d3.interpolate .5, 0
).remove()

# run the mouseon/out functions
circles.on 'mouseover', mouseOn
circles.on 'mouseout', mouseOff

```

```

# tooltips (using jQuery plugin tipsy)
circles.append('title')
  .text (d) -> d.text

$('.circles').tipsy gravity: 's'

circles
  .attr('class', 'tooltipped')
  .attr('data-position', 'bottom')
  .attr('data-delay', '50')
  .attr('data-tooltip', (d) -> d.text)

# the legend color guide
legend = svg.selectAll('rect')
  .data(sentiments)
  .enter()
  .append('rect')
  .attr(
    x: (d, i) -> 40 + i * 80
    y: h
    width: 25
    height: 12)
  .style('fill', (d) ->
    if d == 'Positive' then return color 0.5
    if d == 'Negative' then return color -0.5
    else return color 0
  )

# legend labels
svg.selectAll('text').data(sentiments).enter().append('text').attr(
  x: (d, i) -> 40 + i * 80
  y: h + 24).text (d) -> d

# draw axes and axis labels
svg.append('g')
  .attr('class', 'x axis')
  .attr('transform', "translate(#{margin.l}, #{(h - 60 + margin.t)})")
  .call xAxis

svg.append('g')
  .attr('class', 'y axis')
  .attr('transform', "translate(#{margin.l},#{margin.t})")
  .call yAxis

svg.append('text')
  .attr('class', 'x label')
  .attr('text-anchor', 'end')
  .attr('x', w + 50)
  .attr('y', h - (margin.t) - 5)
  .text 'Frequency'

svg.append('text')
  .attr('class', 'y label')
  .attr('text-anchor', 'end')
  .attr('x', -20)
  .attr('y', 45)
  .attr('dy', '.75em')
  .attr('transform', 'rotate(-90)')
  .text 'Sentiment'

$(document).ready ->
  $('.tooltipped').tooltip delay: 50

```

6.10.37 visualisations/earch-summary-main.coffee

```
hexagonsModule = require '../visualisations/hexagons-module.coffee'

gaugeModule = require '../visualisations/gauge-module.coffee'

pageControls = require '../page-controls-module.coffee'
pageControls.setMainPage 'search'
```

6.10.38 visualisations/text-tweets.coffee

```
makeClickWords = (clickWords, body) ->
  clWord = (word) -> (''+word).toLowerCase().replace /\W/g, ''
  htmlTweet = ''
  aStyle = 'style="color: black; font-weight: bold;" ' # style for hyperlinks
  for word in body.split " "
    if clWord(word) in clickWords
      htmlTweet += "<a #{aStyle} href='/text-tweets/#{clWord word}'>#{word}</a> "
    else htmlTweet += " #{word} "
  htmlTweet

makeDiv = (tweet) ->
  col = if tweet.sentiment > 0 then 'green' else 'darkred'
  html = "<div class='card-panel'"
  html += "<div class='pull-right'"
  html += "<b style='color: #{col}'>#{tweet.sentiment*100}%</b>"
  html += "</div>"
  html += " #{makeClickWords tweet.keywords, tweet.body}"
  if tweet.location.place_name != null
    html += "<br><i class='tiny material-icons small-grey'>location_on</i>"
    html += "<p class='small-grey inline'>#{tweet.location.place_name}</p>"
  html += "<div class='pull-right'"
  html += "<i class='tiny material-icons small-grey inline'>schedule</i>"
  html += "<p class='small-grey inline'>#{tweet.dateTime}</p></div>"
  html += "</div>"
  html

socket = io.connect();

socket.on 'anyTweet', (tweetObj) ->

  searchTerm = $('#txtKeyword').val()

  if (searchTerm == '' or
    (tweetObj.body and tweetObj.body.indexOf(searchTerm) > -1)) and
    $('#showLive').is(':checked')

    if tweetObj.sentiment > 0
      $("#positiveContainer").prepend(makeDiv tweetObj)
      if ($('#positiveContainer .card-panel').length > 100
        $('#positiveContainer .card-panel:last').remove()

    else if tweetObj.sentiment < 0
      $("#negativeContainer").prepend(makeDiv tweetObj)
      if ($('#negativeContainer .card-panel').length > 100
        $('#negativeContainer .card-panel:last').remove()

$('#txtKeyword').keyup (e) -> if e.keyCode == 13
  showLoader()
  window.location = '/text-tweets/'+$('#txtKeyword').val()
```


6.10.39 visualisations/timeline-main.coffee

```
d3 = require 'd3/d3.min.js'
nv = require 'nvd3/build/nv.d3.js'

pageControls = require '../page-controls-module.coffee'
pageControls.setMainPage 'timeline'

nv.addGraph ->

# Initiate NV chart
chart = nv.models.lineChart()
  .margin(left: 100)
  .useInteractiveGuideline(true)
  .showLegend(true)
  .showYAxis(true)
  .showXAxis(true)
  .interpolate("basis")

# Configure x-axis
chart.xAxis
  .axisLabel('Time of Day')
  .tickFormat((d) -> d + ':00')

chart.forceX([7,23])

# Configure y-axis
chart.yAxis
  .axisLabel('Sentiment')
  .tickFormat d3.format('.02f')

# Specify data options
chartData = [
  { values: results.posData, key: 'Positive', color: '#74DF00', area: true }
  { values: results.negData, key: 'Negative', color: '#FA5858', area: true }
]

# Draw chart
d3.select('#chart svg').datum(chartData).call chart

#Update the chart when window re-sizes.
nv.utils.windowResize -> chart.update(); chart
```

6.10.40 visualisations/tone-analysis-main.coffee

```
renderToneBars = require '../visualisations/render-tone-bars-module.coffee'  
renderRadarChart = require '../visualisations/radar-module.coffee'  
  
# Press enter to search  
require('../page-controls-module.coffee').setMainPage 'tone-analyzer'  
  
# Called when server returns results, calls functions to draw all charts  
renderResults = (results) ->  
  renderToneBars results  
  if showAllCharts? && showAllCharts then renderRadarChart results  
  
# Called when page has loaded, initiates the request  
makeRequest = (params) ->  
  $.post('/api/tone', params, (results) -> renderResults results)  
  
$(document).ready ->  
  # Hide empty containers to start with  
  j = 1; while j <= 3 then $('#toneResults'+j).hide(); j++  
  
  # Call function which makes actual request  
  if st? and st != '' then makeRequest { 'searchTerm': st }  
  else if tweetBody? then makeRequest { 'text': tweetBody }
```

6.10.42 visualisations/trending.coffee

```
drawBubbleChart = (results) ->
  diameter = 450
  format = d3.format(',d')

  scaleColors = ["#C80000", "#EB3443", "#B1B1B1", "#42F735", "#4AF43E"]
  color = d3.scale.linear()
  .domain([-0.8, -0.2, 0, 0.2, 0.8])
  .range(scaleColors)

  bubble = d3.layout.pack().sort(null).size([diameter, diameter]).padding(1.5)
  svg = d3.select('#bubble-trends')
  .append('svg')
  .attr('width', diameter)
  .attr('height', diameter)
  .attr('class', 'bubble')

  classes = (root) ->
    classes = []
    recurse = (name, node) ->
      if node.children
        node.children.forEach (child) ->
          recurse node.name, child
      else
        classes.push
          packageName: name
          className: node.name
          value: node.size
          col: node.col
    recurse null, root
    { children: classes }

  root = { name: 'trending', children: [
    { name: 'positive', children: [] }
    { name: 'negative', children: [] }
  ]}

  for trend in results
    if trend.sentiment > 0
      root.children[0].children.push {
        name: trend.topic, size: trend.volume, col: trend.sentiment }
    else if trend.sentiment < 0
      root.children[1].children.push {
        name: trend.topic, size: trend.volume, col: trend.sentiment }

  node = svg.selectAll('.node')
  .data(bubble.nodes(classes(root)))
  .filter((d) -> !d.children)
  .enter()
  .append('g')
  .attr('class', 'node')
  .attr('transform', (d) -> 'translate(' + d.x + ', ' + d.y + ')')

  node.append('title').text (d) -> d.className + ': ' + format(d.value)

  node.append('circle')
  .attr('r', (d) -> d.r)
  .style 'fill', (d) -> color d.col

  node.append('text')
  .attr('dy', '.3em')
  .style('text-anchor', 'middle')
  .text (d) -> d.className.substring 0, d.r / 3

  d3.select(self.frameElement).style 'height', diameter + 'px'
```

```

makeHtmlTrend = (trend) ->
  html = ""
  col =
    if trend.sentiment > 0 then 'green'
    else if trend.sentiment < 0 then 'darkred'
    else 'gray'
  html += "<a href='/search/#{trend.topic.replace(/[\\W_]+/g, '')}>"
  html += "<p class='flow-text center' style='color: #{col}; font-weight: 600>"
  html += "#{trend.topic}"
  html += "</p></a>"
  html

renderResults = (results) ->
  # If we have an error...
  if !results.trends? or results.trends.length < 1
    alert 'That location couldn't be recognised'
    window.location.href = '/trending'
    return

  if $.isEmptyObject(results.trends)
    $('#bubble-trends').append(
      "<h3>No results found for #{results.location}<br>Try another location</h3>"
    )
    $('#trending-text-loader').fadeOut('slow')
    return

  # If everything is all cool then show text trends
  $('#trending-text').hide()
  for t in results.trends then $('#trending-text').append(makeHtmlTrend(t))
  $('#trending-text-loader').fadeOut('fast')
  $('#trending-text').slideDown('slow');
  $('#titleLocation').html(results.location)

  drawBubbleChart(results.trends)

# Make actual data request
$(document).ready ->
  urlPage = window.location.href.split('/').pop().toLowerCase()
  urlEnd = if urlPage != 'trending' then urlPage else ''
  $.post('/api/trending/'+urlEnd, {}, (results) -> renderResults results)

# Execute search query when enter is pressed or button is clicked
$('#txtLocation').bind 'enter', () ->
  window.location.href = '/trending/'+$('#txtLocation').val()
$('#txtLocation').keyup (e) -> if e.keyCode == 13 then $(this).trigger 'enter'
$('#btnSearch').click () ->
  window.location.href = '/trending/'+$('#txtLocation').val()

```

6.10.43 visualisations/word-cloud-main.coffee

```
mainPage = 'word-cloud'
pageControls = require '../page-controls-module.coffee'
pageControls.setMainPage mainPage

WIDTH = 1200 # Width constant for canvas and chart
HEIGHT = 600 # Height constant for canvas and chart

getSentimentForWord = (word) ->
  for each in wordData then if each.text == word then return each.sentiment
  0 # if for some reason we can't find the word, then just return neutral.

scaleColors = ["#a50026", "#d73027", "#f46d43", "#fdae61", "#fee08b", "#B4B4B4",
  "#d9ef8b", "#a6d96a", "#66bd63", "#1a9850", "#006837"]

fillScale = d3.scale.linear()
  .domain([-1, -0.8, -0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.8, 1])
  .range(scaleColors)

sizeScale = d3.scale.linear()
  .domain([0, 80])
  .range([13, 65])

draw = (words) ->
  d3.select('svg#cloud')
    .attr('width', WIDTH)
    .attr('height', HEIGHT)
    .style('padding', HEIGHT/4+'px '+WIDTH/3+'px')
    .append('g')
    .attr('transform', 'translate(150,150)')
    .selectAll('text')
    .data(words)
    .enter()
    .append('text')
    .style('font-size', (d) -> d.size + 'px')
    .style('font-family', 'Impact')
    .style('cursor', 'pointer')
    .style('fill', (d, i) -> fillScale getSentimentForWord d.text)
    .attr('text-anchor', 'middle')
    .attr('transform', (d) ->
      'translate(' + [d.x, d.y,] + ')rotate(' + d.rotate + ')')
    .text(d) -> d.text
    .on('click', (d, i) -> window.location.href = '/word-cloud/'+d.text

d3.layout.cloud().size([WIDTH, HEIGHT])
  .words(wordData)
  .rotate(-> ~ ~ (Math.round(Math.random()*5)*45)-90)
  .font('Impact').fontSize((d) -> sizeScale d.freq)
  .on('end', draw).start()
```

6.10.44 /loading.coffee

```
$(window).load ->
  $('#loading-graphic').fadeOut 'slow'

root = exports ? this
root.showLoader = () -> $('#loading-graphic').fadeIn 'slow'
root.hideLoader = () -> $('#loading-graphic').fadeOut 'slow'

$(document).ready () ->
  $(".button-collapse").sideNav()
  if ($(window).width() <= 699)
    if window.location.pathname.indexOf('mobile') == -1
      document.location = '/mobile'

$('.nav-wrapper ul').click() -> root.showLoader()
```

6.10.45 /page-controls-module.coffee

```
vis = false

if !mainPage then mainPage = 'globe'

$(document).ready ->
  goToUrl = (url) -> showLoader(); window.location = url # Navigate to a URL
  keywordSel = 'input#txtKeyword' # Selector for the keyword search box

  showDetails = () ->
    $('#theContent').slideDown()
    $('#btnHide').text 'Hide'
    true

  hideDetails = () ->
    $('#theContent').slideUp()
    $('#btnHide').text 'Show Details'
    false

  # Submit search term, when the user presses enter
  $(keywordSel).bind 'enter', () -> goToUrl('/'+mainPage+'/'+$(keywordSel).val())
  $(keywordSel).keyup (e) -> if e.keyCode == 13 then $(this).trigger 'enter'
  $('#btnSearch').click () -> goToUrl('/'+mainPage+'/'+$(keywordSel).val())

  $('#btnHide').click () ->
    if vis == true then vis = hideDetails() else vis = showDetails()

module.exports.setMainPage = (val) -> mainPage = val
```

6.11 STYLES (CLIENT-SIDE)

6.11.1 styles/constants.less

```
/*-----*\
*----- COLORS -----*
\******/

@col_darkPrimary: #074B94;
@col_primary: #3E9BFF;
@col_paleprimary: #96c4ff;
@col_white: #fff;
@col_palegrey: rgb(250,250,250);
@col_darkgrey: rgb(175, 175, 175);

/*-----*\
*----- FONTS -----*
\******/

@font_default: "Roboto", "Helvetica Neue", Helvetica, Arial, sans-serif;

/*-----*\
*----- SHADOWS ---*
\******/

@shadow_panel_default: 0 1px 3px rgba(0,0,0,0.15);
@shadow_panel_hover: 0 1px 4px 1px rgba(0,0,0,0.2);
```

6.11.2 styles/general.less

```
@import "constants";

/*-----*\
*-- PAGE LAYOUT -*
\******/

body {
  font: 14px "Lucida Grande", Helvetica, Arial, sans-serif;
  margin: 0;
  background: @col_palegrey;
  display: flex;
  min-height: 100vh;
  flex-direction: column;
}

main {
  flex: 1 0 auto;
  margin: 1em auto;
  padding: 1em;
}

footer{
  padding: 0;
}
```

```

.whitebg{
  background: @col_white;
  box-shadow: 0 1px 3px rgba(0, 0, 0, 0.2);
  &.card-padding{
    padding: 0.5em 1.5em;
  }
  &.curved{
    border-radius: 0.2em;
  }
}

/*****\
*----- TEXT -----*
\*****/
p, a, span, li, html{
  font-family: @font_default;
}

h1, h2, h3, h4, h5{
  font-family: @font_default;
  font-weight: normal;
}

h1{ font-size: 2.5em; }

h2{ font-size: 2.28rem;}

h3{ font-size: 1.8rem; }

/*****\
*-- ADJUSTMENTS--*
\*****/
.absolute{ position: absolute;}
.maxWidth75{ max-width: 75em; }
.zero-margin {margin: 0; }
.zero-padding {padding: 0 !important; }
.pull-right{float:right; }
.pull-left{float:left; }
.auto-width{width: auto !important; }
.center-block {display: block; margin: 0 auto; }
.full-width{width: 100%; }
.small-grey{font-size: 0.8em; color: #7a7a7a; }
.inline{display: inline;}
.block{display: block !important}
.font-size1{font-size: 1em !important;}
.sml-margin{margin: 5px;}
.auto-overflow{overflow: auto; }

```

6.11.3 styles/loading.css

```

#loading-graphic{
  position: fixed;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  z-index: 998;
  background: url(/images/loading.gif) center no-repeat #fff;
}

```


6.11.4 styles/charts.less

```
/* *****\
 * Simple Tweet *
 \******/
.top-tweet{
  min-height: 2em;
  border-bottom: 1px solid #b5b5b5;
  padding: 2px 5px;
  margin-top: 8px;
}

/* *****\
 * Bullet Chart *
 \******/
.bullet { font: 10px sans-serif; }
.bullet .marker { stroke: #29276b; stroke-width: 2px; }
.bullet .tick line { stroke: #666; stroke-width: .5px; }
.bullet .range.s0 { fill: #eee; }
.bullet .range.s1 { fill: #c7c7c7; }
.bullet .range.s2 { fill: #a5a5a5; }
.bullet .measure.s0 { fill: rgba(255, 78, 66, 1); }
.bullet .measure.s1 { fill: #60ff56; }
.bullet .title { font-size: 12px; color: #404040; }
.bullet .subtitle { fill: #999; }

/* *****\
 * Scatter Plot *
 \******/
.axis path,
.axis line {
  fill: none;
  stroke: #000;
  shape-rendering: crispEdges;
  opacity: 1;
}
.axis text { font-size:10px; }
.circles { opacity: .5; }
.tipsy { font-size:11px; margin-top:-10px;}
.guide line {
  fill: none;
  stroke: #000;
  shape-rendering: crispEdges;
  opacity: 0;
}

/* *****\
 * Tree Diagram *
 \******/
#tree {
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  position: relative;
  width: 960px;
  margin: -2em auto 1em auto;
  padding: 0;
}
.tree-node {
  border: solid 1px white;
  font: 10px sans-serif;
  line-height: 12px;
  overflow: hidden;
  position: absolute;
```

```

    text-indent: 2px;
}
.tree-node:hover{ background: grey; }

/*****\
 * Sankey Diagram *
 \*****/
.sankey-node rect {
  cursor: move;
  fill-opacity: .9;
  shape-rendering: crispEdges;
}
.sankey-node text {
  pointer-events: none;
  text-shadow: 0 1px 0 #fff;
}
.sankey-link {
  fill: none;
  stroke: #000;
  stroke-opacity: .2;
}
.sankey-link:hover {
  stroke-opacity: .5;
}

/*****\
 * D3 Tooltip *
 \*****/
.d3-tip {
  line-height: 1;
  font-weight: bold;
  padding: 12px;
  background: rgba(0, 0, 0, 0.8);
  color: #fff;
  border-radius: 2px;
  pointer-events: none;
}
/* Creates a small triangle extender for the tooltip */
.d3-tip:after {
  box-sizing: border-box;
  display: inline;
  font-size: 10px;
  width: 100%;
  line-height: 1;
  color: rgba(0, 0, 0, 0.8);
  position: absolute;
  pointer-events: none;
}
/* Northward tooltips */
.d3-tip.n:after {
  content: "\25BC";
  margin: -1px 0 0 0;
  top: 100%;
  left: 0;
  text-align: center;
}
/* Eastward tooltips */
.d3-tip.e:after {
  content: "\25C0";
  margin: -4px 0 0 0;
  top: 50%;
  left: -8px;
}
/* Southward tooltips */
.d3-tip.s:after {
  content: "\25B2";
  margin: 0 0 1px 0;
}

```

```

    top: -8px;
    left: 0;
    text-align: center;
}
/* Westward tooltips */
.d3-tip.w:after {
  content: "\25B6";
  margin: -4px 0 0 -1px;
  top: 50%;
  left: 100%;
}

/*****\
 * Gauge Diagram *
 \*****/
.c3-chart-arcs-background{fill:#e0e0e0;stroke:none}

/*****\
 * Radar Chart *
 \*****/
.radar-chart .area {
  fill-opacity: 0.7;
}
.radar-chart.focus .area {
  fill-opacity: 0.3;
}
.radar-chart.focus .area.focused {
  fill-opacity: 0.9;
}
.area.germany, .germany .circle {
  fill: #FFD700;
  stroke: none;
}
.area.argentina, .argentina .circle {
  fill: #ADD8E6;
  stroke: none;
}

/*****\
 * Horizontal Bar Chart *
 \*****/
.horizontal-bar{
  height: 15px;
}

#toneResults1 {
  .horizontal-bar:nth-of-type(1) {
    background: #ffaa99;
    .determinate {
      background: #ff694b;
    }
  }
  .horizontal-bar:nth-of-type(2) {
    background: #ffd48f;
    .determinate {
      background: #ffba6c;
    }
  }
  .horizontal-bar:nth-of-type(3) {
    background: #daff90;
    .determinate {
      background: #a5ff41;
    }
  }
  .horizontal-bar:nth-of-type(4) {

```

```

background: #e8c6ff;
.determinate {
  background: #d981ff;
}
}
.horizontal-bar:nth-of-type(5) {
  background: #aeaeff;
.determinate {
  background: #7c7aff;
}
}
}

#toneResults2 {

.horizontal-bar:nth-of-type(1) {
  background: #5bfff5;
.determinate {
  background: #00a2ff;
}
}
.horizontal-bar:nth-of-type(2) {
  background: #fd7aff;
.determinate {
  background: #ff21f4;
}
}
.horizontal-bar:nth-of-type(3) {
  background: #fffa62;
.determinate {
  background: #f8ff00;
}
}
}

/*****\
* Comparison Chart on Search Page *
*****/
.search-av-chart1{
.c3-arc-Positive{ opacity: 0.65 !important; }
.c3-arc-Negative{ opacity: 0.65 !important; }
}

.search-av-chart2{
.c3-arc-Positive{ opacity: 0.35 !important; }
.c3-arc-Negative{ opacity: 0.35 !important; }
}

```

6.11.5 styles/components.less

```
@import "constants";

/*****\
 * Side Nav *
 \*****/
.side-nav{
  padding-top: 65px;

  li{
    height: 2.5em;
    padding: 5px 0 0 5px;
    a{ line-height: 10px; height: 45px; }
  }

  .nav-fixed-bottom{
    position: absolute;
    bottom: 8em;
    height: 8em;
    width: 100%;
  }
}

/*****\
 * Control Options Box *
 \*****/

// The control box super class
.top-left-controls{
  position: absolute;
  margin: 0.5em;
  padding: 0.3em;
  z-index: 999;
  max-width: 20em;
  border-radius: 5px;
  cursor: default;
  opacity: 0.9;
  min-height: 3.5em;
  h2{
    display: inline;
    font-size: 1.5em;
  }
  .outline{
    background: none;
    text-decoration: none;
    border-radius: 4px;
  }
  .outline-button{
    .outline();
    cursor: pointer;
    padding: 0.5em;
    margin: 0.5em;
    width: 7em;
    display: inline-block;
    text-align: center;
    position: relative;
  }
}

// The control box for the globe
.top-left-controls-dark {
  .top-left-controls();
  background: rgba(50, 50, 50, .75);
  h2, p, label{ color: @col_white; }
```

```

.outline-button{
  border: 1px solid @col_white;
  color: @col_white;
  &:hover{ background: rgba(50,50,50,.75)}
}

input{
  width: 15em;
  border: 1px solid @col_white;
  color: @col_white;
  background: none;
  text-decoration: none;
  border-radius: 4px;   margin: 1em 0.5em;
  padding: 0.5em;
  &:focus{ background: rgba(50,50,50,.75); outline: none; }
}

// The control box for the lighter pages
.top-left-controls-light {
  width: 16em;
  padding: 0.3em 0.8em;
  max-height: 550px;
  overflow-y: auto;
  .top-left-controls();
  background: rgba(255,255,255,0.8);
  .outline-button{
    &:hover{ background: rgba(180,180,180,.25)}
  }
  li{list-style-type: none;}
  li a{color: #000000; }
}

/*****\
 * Autocompleter styles *
 \*****/

.autocompleter {
  width: 20em !important;
  z-index: 999;
  background: #ffffff;
}

.autocompleter,
.autocompleter-hint {
  position: absolute;
}

.autocompleter-list {
  padding: 0;
  margin: 0;
  text-align: left;
  list-style: none;
  box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.1);
  -webkit-box-sizing: border-box;
  box-sizing: border-box;
}

.autocompleter-item-selected {
  background: @col_white;
}

.autocompleter-item {
  padding: 6px 12px;
  font-size: 20px;
  color: #444444;
  cursor: pointer;
}

```

```

}

.autocompleter-item:hover {
  background: @col_palegrey;
}

.autocompleter-item strong {
  text-shadow: 0 1px 0 #ffffff;
  background: #b3d4ea;
}

.autocompleter-item span {
  color: @col_palegrey;
}

.autocompleter-hint {
  top: -56px;
  left: 0;
  display: none;
  width: 100%;
  padding: 12px 12px 12px 13px;
  font-size: 24px;
  font-weight: 400;
  color: #efefef;
  text-align: left;
}

.autocompleter-hint span {
  color: transparent;
}

.autocompleter-hint-show {
  display: block;
}

.autocompleter-closed {
  display: none;
}

.autocompleter-item{
  font-size: 1em;
}

.autocompleter{
  margin-top: -15px;
}

```

6.11.6 styles/map.less

```
@import 'constants';

.map-options{
  height: 5em;
  width: 100%;
  background: @col_white;
  box-shadow: @shadow_panel_default;
  margin: 3px 0;

  .input-field{
    input:focus {
      border-bottom-color: @col_primary;
      -webkit-box-shadow: 0 1px 0 0 @col_primary;
      -moz-box-shadow: 0 1px 0 0 @col_primary;
      box-shadow: 0 1px 0 0 @col_primary;
    }

    label:focus{ color: @col_primary; }
  }
}

html, body, #map-canvas {
  height: 100%;
  margin: 0;
  padding: 0;
}

#txtLocation{
  &::-webkit-input-placeholder {
    color: @col_white;
  }
  &::-moz-placeholder { /* Firefox 18- */
    color: @col_white;
  }
  &::-moz-placeholder { /* Firefox 19+ */
    color: @col_white;
  }
  &::-ms-input-placeholder {
    color: @col_white;
  }
}

#btnClearMap{
  margin: 4em 1em;
  padding: 0 0.5em;
  font-size: 0.8em;
}
```


6.11.7 styles/materialize.less

```
@import "constants";
nav, footer.page-footer{
  background: @col_primary;
  z-index: 999;
}

.brand-logo{
  height: 100%
}

.btn, .tabs .indicator{
  background: @col_primary;
  &:hover{ background: @col_paleprimary; }
}

.tabs .tab a{
  color: @col_primary;
  &:hover{ color: @col_darkPrimary; }
}

.flow-text{font-size: 1.5em; }

.card.small{
  margin: 10px;
  padding: 0;
}

.btn-flat:hover{
  background: rgba(200,200,200,0.5);
}
```

6.12 VIEWS (RUN ON SERVER-SIDE, RENDERED ON CLIENT-SIDE)

6.12.1

6.12.2 views/layout.jade

```
block pageConfig
  - var pages = [{title: 'Heat Map', page:'map', index: '3'},
    {title: 'Regional Map', page:'region-map', index: '2'},
    {title: 'Raw Tweets', page:'text-tweets', index: '6'},
    {title: 'Word Cloud', page:'word-cloud', index: '1'},
    {title: 'Comparison', page:'comparer', index: '9'},
    {title: 'Trending Now', page:'trending', index: '10'},
    {title: 'Time Line', page:'timeline', index: '5'},
    {title: '3D Globe', page:'globe', index: '4'},
    {title: 'Entity Extraction', page:'entity-extraction', index: '7'},
    {title: 'Tone Analyzer', page:'tone-analyzer', index: '11'},
    {title: 'Sentiment Search', page:'search', index: '12'} ]

doctype html
html
  head
    title= title
    include ./head
    block head
  body.vertical.layout
    div#loading-graphic
    include ./sidenav
    unless hideNav
      include ./navbar
    block content
    include ./footer_scripts
    block scripts
```

6.12.3 views/error.jade

```
extends layout

block content
  main.content.error-page
    h3 We have a slight problem...
    h2= error.status
    h1= message
    a.waves-effect.waves-light.btn.center-block(href='/') Head back to the homepage

include ./footer
```

6.12.4 views/component-divider.jade

```
div.col.s12
  br
  div.divider
  br
```

6.12.5 views/component-spinner.jade

```
.preloader-wrapper.big.active
  .spinner-layer.spinner-blue
    .circle-clipper.left
      .circle
    .gap-patch
      .circle
    .circle-clipper.right
      .circle
  .spinner-layer.spinner-red
    .circle-clipper.left
      .circle
    .gap-patch
      .circle
    .circle-clipper.right
      .circle
  .spinner-layer.spinner-yellow
    .circle-clipper.left
      .circle
    .gap-patch
      .circle
    .circle-clipper.right
      .circle
  .spinner-layer.spinner-green
    .circle-clipper.left
      .circle
    .gap-patch
      .circle
    .circle-clipper.right
      .circle
```

6.12.6 views/footer.jade

```
footer.page-footer.zero-padding
  .container
    .row
      .col.l6.s12
        a(href='/'): h5.white-text Sentiment Sweep
        p.grey-text.text-lighten-4 The ultimate social media sentiment analysis
      .col.s3
        ul
          li: b: a.grey-text.text-lighten-3(href='/') Main Menu
          each page in pages.splice(0,7)
            li: a.grey-text.text-lighten-3(href='/#{page.page}') = page.title
      .col.s3
        ul
          each page in pages.splice(0,6)
            li: a.grey-text.text-lighten-3(href='/#{page.page}') = page.title
            li: a.grey-text.text-lighten-3(href='/break-down') Topic Breakdown
            li: a.grey-text.text-lighten-3(href='/hexagons') Sentiment Hexagons
            li: a.grey-text.text-lighten-3(href='/word-scatter-plot') Word Scatter
      .col.s3
        Plot
        Comparison
        li: a.grey-text.text-lighten-3(href='/sa-comparison') SA Method

    .footer-copyright
      .container
        a.flow-text.white-text(href='http://aliciaspykes.com') &copy; Alicia Sykes
        2016
        a.flow-text.white-text.right(href='https://github.com/Lissy93/twitter-sentiment-visualisation') Source Code & Documentation
```

6.12.7 views/footer-scripts.jade

```
script(type='text/javascript',
src='/bower_components/jquery/dist/jquery.min.js')

script(type='text/javascript',
src='/bower_components/materialize/dist/js/materialize.min.js')

script(type='text/javascript',
src='/javascripts/loading.js')
```

6.12.8 views/head.jade

```
link(rel='stylesheet',
href='/bower_components/materialize/dist/css/materialize.min.css')

link(rel='stylesheet',
href='/stylesheets/all.min.css')

link(rel='stylesheet',
href='https://fonts.googleapis.com/icon?family=Material+Icons',)
```

6.12.9 views/side-nav.jade

```
#slide-out.side-nav
  ul
    li: a(href='/') Home Page
    each page in pages
      li: a(href='/#{page.page}') = page.title

.nav-fixed-bottom
  ul
    li: a(href='/', target='_blank') Home
    li: a(href='/about', target='_blank') About
    li: a(href='https://github.com/Lissy93/twitter-sentiment-
visualisation', target='_blank') Source Code & Documentation
    li: a(href='http://aliciasnykes.com', target='_blank') &copy; Alicia
Sykes 2016
```

6.12.10 views/navbar.jade

```
nav
  .nav-wrapper
    a.button-collapse(href='#', data-activates='slide-out')
      i.medium.material-icons reorder
    a.brand-logo.waves-effect.waves-block.waves-light(href='/')
      h1.title Sentiment Sweep
    ul#nav-mobile.right.hide-on-med-and-down.waves-effect.waves-block.waves-
light
      li(class=(pageNum===0 ? 'active' : ''))
        a(href='/') Dashboard
      li(class=(pageNum===1 ? 'active' : ''))
        a(href='/map') Heat Map
      li(class=(pageNum === 6 ? 'active' : ''))
        a(href='/region-map') Region Map
      li(class=(pageNum === 4 ? 'active' : ''))
        a(href='/globe') Globe
      li(class=(pageNum === 10 ? 'active' : ''))
        a(href='/comparer') Comparer
      li(class=(pageNum===3 ? 'active' : ''))
        a(href='/timeline') Timeline
      li(class=(pageNum===5 ? 'active' : ''))
```

```

      a(href='/word-cloud') Word Cloud
    li(class=(pageNum===7 ? 'active' : ''))
      a(href='/text-tweets') Raw Tweets
    li(class=(pageNum===8 ? 'active' : ''))
      a(href='/entity-extraction') Entity Extraction

```

6.12.11 index.jade

extends layout

block head

```

  link(href='https://fonts.googleapis.com/icon?family=Material+Icons',
rel='stylesheet')

```

block pageConfig

```

- var hideNav = true

```

block content

```

  a#github-corner(href='https://github.com/Lissy93/twitter-sentiment-
visualisation', target = '_blank')
    img(src='/images/github-corner.png', title='View source code and
documentation on GitHub')

  #part-1
    .row.absolute.home-row
      .col.s6
        .card-panel
          h3 Introduction to Sentiment Sweep
          p Sentiment Sweep aims to captcha the mood of the internet, either
            | overall or towards a specific topic.
            | It does this by analysing real-time Twitter data, and calculating
            | how positive or negative each Tweet it.

          p A series of dynamic data visualisations are then used to
            | illustrate the results, and find trends between sentiment and
            | other factors such as time of day, location, topic, country,
people etc

          p It's open sauce, and the code and documentation can be viewed
            | on <a href='https://github.com/Lissy93/twitter-sentiment-
visualisation'>GitHub</a>

          a.waves-effect.waves-teal.btn-flat.pull-right(href='/about') Read
More...

        .col.s6
          .card-panel
            h3 Get started, enter a search term...
            .input-field
              input#txtKeyword.validate(type='text')
              label(for='txtKeyword') Enter a keyword, topic, brand, celebrity or
search term

            h3 ... or scroll down to view more data visualisations

          #chart.home-hexagons

          a.scroll-down#scroll-1(href='#')
            i.material-icons.large.center-align play_for_work
            p.center-align Scroll Down

          a#hex-details(href='/hexagons')
            p The hexagon background is made up of Tweets from the past 60 seconds.
              | Hover over a hexagon to read the associated Tweet.
              | Sentiment is represented by color

  #part-0
    p

```

```

.section#part-2
  .container
    h2 Sentiment Data Visualisations
    p.flow-text.grey-text.darken-4 Click one of the links below to generate
the chart with the latest Twitter data
    .row
      each page in pages
        .col.s3
          a(href='/#{page.page}')
            .card.home-card
              .card-image.waves-effect.waves-block.waves-light
                img(src='/images/thumbnails/thumb_#{page.index}.png')
              .card-content
                span.card-title.grey-text.text-darken-4= page.title

    footer
      .footer-copyright
        .row
          .col.s3
            p
          .col.s2
            a.small-grey.right(href='/about') About Sentiment Sweep
          .col.s2
            a.small-grey.right(href='https://github.com/Lissy93/twitter-
sentiment-visualisation') Source Code and Documentation
          .col.s2
            a.small-grey.right(href='http://aliciasykes.com') &copy; Alicia
Sykes 2016

    block scripts
      script.
        var homePage = true; // Show different hexagons
        var results = !JSON.stringify(data); // Get and parse Twitter results
        var average = !JSON.stringify(averageSentiment); // Average sentiment

        script(type = 'text/javascript', src = 'https://cdn.socket.io/socket.io-
1.4.3.js')
        script(type = 'text/javascript', src = '/javascripts/charts/home-page.js')
        script(type = 'text/javascript', src = '/bower_components/d3/d3.min.js')
        script(type = 'text/javascript', src = '/bower_components/hexbin/index.js')
        script(type = 'text/javascript', src = '/bower_components/d3tip/index.js')
        script(type='text/javascript', src='/javascripts/charts/hexagons-module.js')

```

6.12.12 views/page-about.jade

```
extends layout
block content
  main.content.about-page
    h1 About Sentiment Sweep

    h2 Introduction
    .row.card-panel
      h3 What is Sentiment Sweep?
      p.flow-text
        | Sentiment Sweep aims to capture the mood of the internet,
        | either overall or towards a specific topic.
        | It does this by analysing real-time Twitter data, and
        | calculating how positive or negative each Tweet is.
      p.flow-text
        | Ten dynamic data visualisations are then used to illustrate
        | the results, and find trends between sentiment and other factors
        | such as time of day, location, topic, country, people etc

      h3 What can it be used for?
      p.flow-text
        | SS has many uses, from analysing how successful a marketing
        | campaign was in various locations and times, and comparing it
        | to your competitors, to predicting how stock prices will change
        | based on what people are saying about that company. It can also
        | be used to predict which political party will be next in power
        | before the polls are in, based on popularity of each.
        | It can show which train lines are currently delayed or overcrowded,
        | or which the football team has the most support at that time,
        | to name just a few uses.
      h3 Is it free?
      p.flow-text Yes.

    h2 Technical Overview
    .row.card-panel
      h3 How does it work?
      p.flow-text Well, in short Tweets are streamed live from Twitter, and
        | as each one comes in an algorithm calculates the sentiment based on
        | which words are in the Tweet and how they are positioned.
        | A function is then triggered that formats that Tweet and adds it
        | to whichever data visualisations are currently being viewed.
        | If you search for a custom search term, then that will fetch fresh
        | data from Twitter relative to what you searched and the synonyms.
        | If it's for a location based visualisation that the Google Places API
        | is used in conjunction to get the latitude and longitude of where the
        | Tweet was from. Data from the last 60 minutes is cached, so that
        | everything loads fast.
      p.flow-text All the data visualisations use the same backend, but have
        | different adapter code to format the data into the right format for
        | each chart, most of which is done server-side for efficiency.
        | Everything's written in a very modular way, for maximum code
reusability.

      h3 Modular code, what's that?
      p.flow-text
        | Well most the components of sentiment sweep are quite generic, so
        | could be used for other things in other programs. For this reason
        | eight node modules have been written, documented and tested then
        | published free under the MIT license on NPM. They are in separate
        | repositories, but you can find links to them in the readme.md

      h3 So what's it written in?
      p.flow-text The backend is mainly Node.js written in CoffeeScript, and
        | using MongoDB for data caching. The data visualisations are mainly
        | developed using D3.js and most of them are rendered isomorphically
        | so that real-time changes are efficient, again these are written in
        | CoffeeScript. The Express framework is used for routing, and page
        | layouts are written in Jade, with styles written in LESS.
```

```

h3 Nice, so can I see the code?
p.flow-text Yes, because it's 100% open sauce! Everything is published on
| GitHub under the MIT license, so you can view and copy whatever
| you want :)
a(href='https://github.com/Lissy93/twitter-sentiment-visualisation')
| Check it out here

h3 How will I know what's what in the code?
p.flow-text There is a lot of code, but it's all fully documented both
| in-code documentation, unit tests, and written documentation.
a(href='https://github.com/Lissy93/twitter-sentiment-
visualisation/tree/dev/docs')
| You can read the written documentation here

h2 Stability
.row.card-panel
h3 How do you know this works?
p.flow-text It's unit tested.

h3 What if I find a bug?
p.flow-text If you find something that doesn't work as you'd expect,
| then you can raise it as an issue
a(href='https://github.com/Lissy93/twitter-sentiment-
visualisation/issues')
| here.
h3 The live data has stopped working, why?
p.flow-text Probably the API limits have been reached, try again tomorrow.
| This will be sorted later on when we can qualify for a larger quota.

h3 What if I think this could be better?
p.flow-text I'm always open to suggestions, there are contact details at
the bottom of the page.
| Or create a pull request on GitHub and implement your modifications
for review.

h2 Legal
.row.card-panel
h3 So I can copy the code, right?
p.flow-text Yeah, because it's open-source. But you can't duplicate this
| project and pass it of as your own work.

h3 Is it guaranteed to work?
p.flow-text Nope, no guarantees, use at your own risk.

h3 Is it okay to be using people's Tweets like this?
p.flow-text Sentiment Sweep follows the rules outlined in
| the Twitter API T&C's. Only public Tweets are displayed,
| and the users location is blurred by 100m, to hide their exact
address.
| No Tweets are cached for more than 24 hours (usually it's less than
an hour)

h2 License
.row.card-panel
h3 MIT License
p.flow-text Copyright (c) Alicia Sykes <alicia@aliciasykes.com>
p.flow-text
| Permission is hereby granted, free of charge, to any person obtaining
a copy
| of this software and associated documentation files (the "Software"),
to deal
| in the Software without restriction, including without limitation the
rights
| to use, copy, modify, merge, publish, distribute, sub-license, and/or
sell
| copies of the Software, and to permit persons to whom the Software is
furnished
| to do so, subject to the following conditions:

```



```
    p.flow-text
    | The above copyright notice and this permission notice shall be
included install
    | copies or substantial portions of the Software.
    p.flow-text
    | THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR
    | IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANT
ABILITY,
    | FITNESS FOR A PARTICULAR PURPOSE AND NON INFRINGEMENT. IN NO EVENT
SHALL
    | THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER
    | LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM,
    | OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

h2 Author

```
.row.card-panel
```

```
  h3 Who wrote this thing?
```

```
  p.flow-text
```

```
    | Me, Alicia Sykes.
```

```
  h3 Why did you make it?
```

```
  p.flow-text
```

```
    | Thought it would be cool
```

```
  h3 Did it take long to develop?
```

```
  p.flow-text
```

```
    | Yes, ages - but I like coding :)
```

```
  h3 How can I contact you?
```

```
  p.flow-text
```

```
    | Visit my website, there's a contact form:
    a(href='http://aliciaspykes.com') http://aliciaspykes.com
```

```
  h3 Are you available for hire?
```

```
  p.flow-text
```

```
    | No, I get a lot of emails asking this. I am not looking for work at
this time!
```

```
include ./footer
```

6.12.13 views/page-breakdown.jade

extends layout

block head

```
link(href='https://fonts.googleapis.com/icon?family=Material+Icons',
rel='stylesheet')
```

block content

```
div.row.center-block(style='width: 80%')
  div.col.s9.card-panel.input-field(style='height: 4em')
    input#txtKeyword.validate(type='text', value='#{searchTerm}')
    label(for='txtKeyword', style='margin-top: 1em') Enter a keyword, topic
or search term to fetch new Tweets

  div.col.s2.offset-s1.card-panel(style='height: 4em')
    form.tree
      input#size(type='radio', name='mode', value='size', checked='')
      label(for='size') Size
      input#count(type='radio', name='mode', value='count')
      label(for='count') Count

div.row.center-block(style='width: 80%').card-panel
  if searchTerm
    - var relatingTo = ' relating to <b>'+searchTerm+'</b>'
  else
    - var relatingTo = ''
  p The block diagram shows key terms that represented
sentiment!{relatingTo}.
  | The color represents sentiment, and the size of the block, weighting.
  b You can hover over the blocks to see the topic being described.
  div#tree
  br

block scripts
script(type='text/javascript').
  var data = !{JSON.stringify(data)};
script(type='text/javascript', src='/bower_components/d3/d3.min.js')
script(type='text/javascript', src='/javascripts/charts/break-down.js')
```

6.12.14 views/page-cloud.jade

```
extends layout
block content
  div.top-left-controls-light
    .row
      .col.s12
        ul.tabs
          li.tab.col.s3
            a.active(href='/word-cloud') Cloud
          li.tab.col.s3(onClick='showLoader(); window.location="/word-
scatter-plot/#{summary_text.searchTerm}"/')
            a(href='/word-scatter-plot') Scatter

        h2 Intro
        p Word cloud shows keywords commonly used in the latest Tweets. <br>
          | The colour indicates sentiment of the containing Tweet
          | (red= negative, and green= positive),
          | and the size of the word represents the frequency of use.
        h2 Custom Twitter Data
        br
        label.active(for='txtKeyword') Enter Keyword (press Enter to search)
        input#txtKeyword.validate(type='text', value='#{summary_text.searchTerm}')
        br
        div.waves-effect.waves-light.btn.centerBlock#btnHide Show More
        br
        div#theContent(style='display:none')
          h2 Top Words used in Positive Tweets
          br
          ul
            each val in summary_text.topWords.topPositive
              li
                a(href='/word-cloud/#{val.text}') #{val.freq} x
                <b>#{val.text}</b> <span style='color: green'>#{val.sentiment*100}%
                positive</span>
              br
          h2 Top Words used in Negative Tweets
          br
          ul
            each val in summary_text.topWords.topNegative
              li
                a(href='/word-cloud/#{val.text}') #{val.freq} x
                <b>#{val.text}</b> <span style='color: red'>#{val.sentiment*100}%
                negative</span>
              br
          if summary_text.trending
            h2 Trending Key Words
            br
            ul
              each val in summary_text.trending
                li
                  a(href='/word-cloud/#{val.text}') #{val.freq} x
                <b>#{val.text}</b>
            br
          else
            a.waves-effect.waves-light.btn.centerBlock(href='/scatter-plot-
words') View on Scatter Plot
            br
            br

        main.content.zero-padding.zero-margin
          svg#cloud(style='display: block; margin: 0 auto')

block scripts
  script(type='text/javascript', src='/bower_components/d3/d3.min.js')
  script(type='text/javascript', src='/bower_components/d3-word-cloud/index.js')
  script(type='text/javascript').
    var wordData = !JSON.stringify(data));
  script(type='text/javascript', src='/javascripts/word-cloud.bundle.js')
```

6.12.15 views/page-comparer.jade

extends layout

block content

```
.container
  if data != ''
    .row
      for result, index in data
        - var topic = result.searchTerm.charAt(0).toUpperCase() +
result.searchTerm.slice(1)
        div(class='col s#{12/data.length}')
          h3= topic
          .card-panel
            p The overall average sentiment for <i>#{topic}</i>
is:
            if result.averageSentiment > 0
              p(style='color: green; font-size: 1.5em') Positive
(<b>#{Math.round(result.averageSentiment * 100)}%</b>)
            else if result.averageSentiment < 0
              p(style='color: darkred; font-size: 1.5em')
Negative (<b>#{Math.round(result.averageSentiment * -100)}%</b>)
            else
              p(style='color: grey; font-size: 1.5em') Neutral
          .card-panel
            h4.flow-text Summary of Tweets for <i>#{topic}</i>
            div(id='chart-#{index}')
          .card-panel
            h4.flow-text Top words used in <i>#{topic}</i> Tweets
            ol
              for wordObject in result.keywordData
                if wordObject.sentiment > 0
                  - var color = 'green'
                else if wordObject.sentiment < 0
                  - var color = 'darkred'
                else
                  - var color = 'dimgrey'
                li
                  a(href='/search/#{wordObject.text}',
style='color: #{color}; font-size: 1.1em; font-weight: bold;') #{wordObject.text}
          .card-panel
            h4.flow-text View more data visualisations for
<i>#{topic}</i>
            a.waves-effect.waves-teal.btn-flat.block(href='/word-
cloud/#{topic}') Word Cloud
            a.waves-effect.waves-teal.btn-
flat.block(href='/regional-map/#{topic}') Regional Map
            a.waves-effect.waves-teal.btn-
flat.block(href='/map/#{topic}') Heat Map
            a.waves-effect.waves-teal.btn-
flat.block(href='/globe/#{topic}') 3D Globe
            a.waves-effect.waves-teal.btn-
flat.block(href='/timeline/#{topic}') TimeLine
            a.waves-effect.waves-teal.btn-flat.block(href='/text-
tweets/#{topic}') Raw Tweets
            a.waves-effect.waves-teal.btn-flat.block(href='/entity-
extraction/#{topic}') Entity Extraction
            a.waves-effect.waves-teal.btn-flat.block(href='/break-
down/#{topic}') Topic Breakdown
    .row
      br
```

```

br
.center-block(style='width: 50%').card-panel
  h2 Brand Comparison
  span.flow-text Enter up to four brand names, retailers, businesses,
    | places, celebrities or objects that you'd like to compare
    | peoples opinions on.
br

.input-field
  label.active(for='brand-1') First Search Term
  input.validate#brand-1 (type='text')

.input-field
  label.active(for='brand-2') Second Search Term
  input.validate#brand-2 (type='text')

.input-field(style='display:none')
  label.active(for='brand-3') Third Search Term
  input.validate#brand-3 (type='text')

.input-field(style='display: none')
  label.active(for='brand-4') Forth Search Term
  input.validate#brand-4 (type='text')

a.small#add-new (style='cursor: pointer') Add more search queries
br
br

.waves-effect.waves-light.btn.center-block#get-results Get Results!

```

block scripts

```

script (type = 'text/javascript', src = '/bower_components/d3/d3.min.js')
script (type='text/javascript', src='/bower_components/c3/c3.min.js')
script (type='text/javascript', src='/javascripts/charts/comparer.js')
script (type='text/javascript').
  window.drawDonuts(!JSON.stringify(data));

```

6.12.16 views/page-sa-comparison.jade

```
extends layout
block content
  main.content.maxWidth75
    div.whitebg.card-padding.curved
      h2 Comparison of methods of calculating Sentiment Analysis
      p
      | There are a number of different methods of calculating
      | sentiment analysis (SA). For this experiment I compare the
      | two most common methods, dictionary-based SA and natural
      | language understanding SA.
      | A set of Tweets will be fetched from the Twitter API, and
      | a script will calculate the sentiment for with both methods
      | and display the results in the table and charts below.
    div.row
      include ./component_divider
      div.col.s12.m6
        h3 Dictionary-based SA
        p
        | Dictionary-based sentiment analysis uses predefined dataset
        | of words annotated with their semantic values. The
algorithm
        | simply looks up the semantic value for each word and then
        | calculates the overall sentiment for the sentence based on
        | the values for each word. <br />
        | The word list used in this example is the AFINN-111.
        | You can view the sourcecode and documentation for the
        | dictionary-based SA module
        | <a href="https://github.com/Lissy93/sentiment-
analysis">here</a>
        div.col.s12.m6
          h3 Natural Language Understanding SA
          p
          | Natural language understanding-based sentiment analysis is a
          | little more complex, and takes longer for each calculation. It
          | also tends to be more costly as well, however the results are
          | considerably more accurate. It is also possible to extract
          | more than just a positivity value for each string, you can
          | drill down into much more detail about the attitudes conveyed.
          | <br />
          | For this example the HP Haven OnDemand API has been used
        include ./component_divider
        div.col.s12.m6(style='float:right')
          h3 Data Sauce
          if(searchTerm)
            p Showing results related to #{searchTerm}
          else
            p
            | The following examples are using Twitter data relating to the
            | Edward Snowden news story in 2013. There is also a third
            | column added which represents the sentiment calculated by a
            | number of humans in a questionnaire, this is to act as a
            | benchmark <br>
            | You can enter your own search query below, Twitter data will
            | be fetched and the sentiment of each Tweet then calculated.
        div.col.s12.m6
          h3 Custom Data Sauce
          input#txtKeyword(
placeholder='Enter keyword to calculate custom sentiment comparison data',
type='text', class='validate', value="#{searchTerm}")
          a#btnCalculate.waves-effect.waves-light.btn(href='',
style='float:right', onclick='showLoader()') Calculate
```

```

include ./component_divider

div.row
  div.col.s12.m6
    div.whitebg.card-padding.curved.card-margin(style='width:35em')
      span#summarry_bar_ch

    div.col.s12.m6
      img(src='/images/sentiment-comparison.png',
width='500px').whitebg.card-padding.curved

div.row.whitebg.card-padding.curved
  span#scatter_ch

div.row.whitebg.card-padding.curved
  table#table_ch.bordered.centered.responsive-table(style='width:71em')

div.row.whitebg.card-padding.curved
  span#bar_ch

script(type='text/javascript', src='https://www.google.com/jsapi')
script(type='text/javascript').
  var sentimentResults = !JSON.stringify(results));
script(type = 'text/javascript', src =
'/bower_components/d3/d3.min.js')
script(type='text/javascript', src='/javascripts/charts/comparison.js')

include ./footer

```

6.12.17 views/page-entity-extraction.jade

extends layout

block head

```

link(href='https://fonts.googleapis.com/icon?family=Material+Icons',
rel='stylesheet')

```

block content

```

div.container
  .row
    div.col.s8.offset-s2.card-panel
      h3 Entity Extraction
      p Extract key information (people, places, topics, companies,
media...) from Tweets.
      | Enter a search term to fetch and analyse fresh Twitter data.
      div.input-field
        input#txtKeyword.validate(type='text', value='#{searchTerm}')
        label(for='txtKeyword') Enter a keyword, topic or search term
to fetch new Tweets

    .row
      div.col.s12.card-panel
        p#chart

div.container(style='width: 90%')
  for key in Object.keys(data)
    h4 #{key.charAt(0).toUpperCase()+key.split('_')[0].slice(1)}
    div.row
      for entity in data[key]
        div.card.small.col.s2
          div.card-image

```

```

        img.better-img-card-
position(src='#{entity.additional_information.image}')
        span.card-title #{entity.normalized_text}
    div.card-content
        p <b>#{entity.matches.length}</b> occurrences found
        for match in entity.matches.splice(0,4)
            span= ' '+match+' '
            if entity.matches.length > 1
                span ...
    div.card-action
        if entity.additional_information.wiki != ''
            a(href='#{entity.additional_information.wiki}')
Wikipedia
            a(href='/entity-extraction/'+entity.normalized_text)
Entities

block scripts
    script(type='text/javascript').
        var results = !{JSON.stringify(data)};
        var searchTerm = !{JSON.stringify(searchTerm)};
        var sankeyData = !{JSON.stringify(sankeyData)};
    script(type='text/javascript', src='/bower_components/d3/d3.min.js')
    script(type='text/javascript', src='/bower_components/sankey.js/index.js')
    script(type='text/javascript', src='/javascripts/charts/entity-extraction.js')

```

6.12.18 views/page-word-plot.jade

```

extends layout
block content
    div.top-left-controls-light.card-panel(style='width: 13em')
        .row
            .col.s12
                ul.tabs
                    li.tab.col.s3(onClick='showLoader(); window.location="/word-
cloud/#{summary_text.searchTerm}"')
                        a(href='/word-cloud') Cloud
                    li.tab.col.s3
                        a.active(href='/word-scatter-plot') Scatter
                h2 Intro
                p Scatter chart showing keywords commonly used in the latest Tweets, along
                | with their sentiment, and frequency of use.
                br
                h2 Custom Data
                br
                label.active(for='txtKeyword') Enter Keyword
                input#txtKeyword.validate(type='text',
value='#{summary_text.searchTerm}')
                br
                div#scatter-words.center-block.card-panel

block scripts
    script(type='text/javascript').
        var wordData = !{JSON.stringify(data)};
    script(type='text/javascript', src='/javascripts/scatter-words.bundle.js')

```


6.12.19 views/page-globe.jade

```
extends layout
block pageConfig
  - var hideNav = true

block content
  div.top-left-controls-dark
    a.outline-button.auto-width(href='/', title='Back to Home') &#8592; Home
    h2 Sentiment Globe
    input#txtKeyword(placeholder='Enter a keyword or topic',
value='#{summary_text.searchTerm}')
    div.outline-button#btnSearch Search
    div.outline-button(title='Hide this container', id='btnHide') More Data

    div#theContent(style='display:none')
      p The globe shows the sentiment from Tweets around the world,
        | in real-time. Red indicates negative and green positive.
        | The height of the bar represents the scale of the sentiment.
        | You can use your mouse to rotate the globe.
      br
      p!=summary_text.globeSentence
      br
      a.outline-button(href='/text-tweets/#{summary_text.searchTerm}') Raw
Tweets
      a.outline-button(href='/word-cloud/#{summary_text.searchTerm}') Key
Words
      a.outline-button(href='/map/#{summary_text.searchTerm}') Heat Map
      a.outline-button(href='/region-map/#{summary_text.searchTerm}') Region
Map

    div#container

block scripts
  script(type='text/javascript', src='/bower_components/Detector/index.js')
  script(type='text/javascript', src='/bower_components/three.min/index.js')
  script(type='text/javascript', src='/bower_components/Tween/index.js')
  script(type='text/javascript', src='/bower_components/globe/index.js')
  script(type='text/javascript', src='https://cdn.socket.io/socket.io-1.4.3.js')
  script(type='text/javascript').
    var sentimentResults = !{JSON.stringify(data)};
  script(type='text/javascript', src='/javascripts/globe.bundle.js')
```

6.12.20 views/page-hexagons.jade

```
extends layout

block content
  br
  span.grey-text
    | Hover over a hexagon to read the associated Tweet.
    | Sentiment is represented by color.
    | All Tweets are real-time and from the past 60 seconds.
  #chart(style='width: 100%; margin: 0; padding: 0;').center-block

block scripts
  script(type='text/javascript').
    var results = !{JSON.stringify(data)};
    var average = !{JSON.stringify(averageSentiment)};
    var searchTerm = !{JSON.stringify(searchTerm)};
    var hexPage = true;
  script(type='text/javascript', src='https://cdn.socket.io/socket.io-
1.4.3.js')
  script(type='text/javascript', src='/bower_components/d3/d3.min.js')
  script(type='text/javascript', src='/bower_components/d3tip/index.js')
  script(type='text/javascript', src='/javascripts/charts/hexagons-module.js')
```

6.12.21 views/page-realtime.jade

extends layout

block head

```
link(href='/bower_components/nvd3/build/nv.d3.min.css', rel='stylesheet')
```

block content

```
.container(style='width: 90%')
  .card-panel#real-time-scatter: svg(style='width: 100%; height: 500px;')
  #scatter-loader.center-block(style='width: 50px; margin-top: 200px'):
include ./component_spinner
```

block scripts

```
script(type = 'text/javascript', src = 'https://cdn.socket.io/socket.io-
1.4.3.js')
script(type = 'text/javascript', src = 'http://nvd3.org/assets/lib/d3.v3.js')
script(type='text/javascript', src='http://nvd3.org/assets/js/nv.d3.js')
script(type='text/javascript', src='/javascripts/charts/real-time-dash.js')
```

6.12.22 views/page-realtime.jade

extends layout

block head

```
link(rel='stylesheet', href='/bower_components/nvd3/build/nv.d3.min.css')
```

block content

```
main.content.timeline.full-width
  div(style='width: 80%; margin-bottom: 1em').center-block
    div.row
      div.col.s6.card-panel(style='min-height: 10em')
        h3 #{{searchTerm}} Sentiment Timeline
        p.flow-text Timeline shows the average positive and negative
sentiments of
          | Tweets over the last 24 hours.
          br
      div.col.s1
        br

      div.col.s5.card-panel(style='min-height: 10em')
        p.flow-text Enter a keyword or topic to view the sentiment over
time
          label.active(for='txtKeyword', style='width: 8em') Enter
keyword or topic to fetch new Tweets
          input#txtKeyword.validate(type='text', style='width: 12em;
margin-right: 1em', value='#{{searchTerm}}')
          div.btn#btnSearch Search
          br

    div#chart.card-panel(style='width: 80%; height: 25em;').center-block
      svg

block scripts
  script(type='text/javascript').
    var results = !{JSON.stringify(data)};
  script(type='text/javascript', src='/bower_components/d3/d3.min.js')
  script(type='text/javascript', src='/javascripts/timeline.bundle.js')
```

6.12.23 views/page-trending.jade

extends layout

```
block content
  .container(style='width: 60%')
    .row.card-panel
      h3 Search trends by location
      .col.s8
        .input-field
          input#txtLocation.validate(type='text', value='#{location}')
          label(for='txtLocation') Enter a location to fetch local

Twitter trends
  .col.s4
    br
    a.btn.waves-effect.waves-light#btnSearch Find Trending Topics
  .row.card-panel
    if location == ''
      h3 Showing world-wide Twitter trends
    else
      h3 Showing the latest Twitter trends from <span
id='titleLocation'>#{location}</span>
      p.small-grey Sorted by volume, and the color indicates the sentiment of
associated Tweets.
      | Click a trend for more information
      #trending-text-loader.center-block(style='width: 2px;'): include
./component_spinner
      #trending-text

  .row.card-panel
    #bubble-trends.center-block(style='width: 460px;')
    p.flow-text Bubble chart shows trending topics for today from a
specific location.
      | The size of the bubble represents the volume of Tweets about a
certain
      | topic, and the color illustrates sentiment calculated from Tweets
      | associated with the topic. Hover over a bubble for more details.

block scripts
  script(type='text/javascript').
    var locationTxt = !{JSON.stringify(location)};
  script(type='text/javascript', src='/bower_components/d3/d3.min.js')
  script(type='text/javascript', src='/javascripts/charts/trending-bubble-
chart.js')
  script(type='text/javascript', src='/javascripts/charts/trending.js')
```

6.13 DEVELOPMENT AND BUILD FILES

6.13.1 ./gulpfile.js

```
/**
 * @author Alicia Sykes <alicia@aliciasykes.com> June 2015
 * To run script run "gulp" in the command line
 * My super amazing gulp setup, does EVERYTHING cool
 * possible to do to turns already awesome code into
 * even more awesomely efficient code
 *
 * The Gulp tasks are divided into the files in the './tasks/' directory
 * Read the build documentation at './docs/building.md'
 */

// Include gulp and require directory module
var gulp = require('gulp');
var reqdir = require('require-dir');

reqdir('./tasks'); // Include the folders containing ALL gulp tasks

gulp.task('default', ['start']); // Default task
```

6.13.2 tasks/browser-sync.js

```
/* Include the necessary modules */
var gulp = require('gulp');
var bSync = require('browser-sync');
var runSequence = require('run-sequence');

var CONFIG = require('../tasks/config');

gulp.task('start', function () {
  runSequence('build', 'test', 'nodemon', 'browser-sync');
});

gulp.task('browser-sync', function () {
  bSync.init({
    files: [CONFIG.SOURCE_ROOT+'/**/*.*.'],
    proxy: 'http://localhost:3000',
    port: 4000,
    browser: ['google chrome'],
    open: false // Don't open browser automatically - it's annoying
  });

  /* For each CoffeeScript directory, watch, compile and reload */
  CONFIG.SCRIPT_PATHS.forEach(function(path) {
    gulp.watch(path.src, ['scripts']).on('change', bSync.reload);
  });

  /* Watch build and re-bundle browserify files */
  gulp.watch(CONFIG.SOURCE_ROOT+'/**/*-{main,module}.{js,coffee}',
    ['browserify'])
    .on('change', bSync.reload);

  /* Watch compile and reload LESS, SASS and CSS */
  gulp.watch(CONFIG.SOURCE_ROOT+'/**/*.{css,less}', ['styles']).on('change',
    bSync.reload);

  /* And reload browsers when the views change too */
  gulp.watch("views/**/*.jade").on('change', bSync.reload);
});
```

6.13.3 tasks/browserify.js

```
var gulp = require('gulp');
var browserify = require('browserify');
var source = require('vinyl-source-stream');
var coffeeify = require('coffeeify');
var reactify = require('reactify');
var buffer = require('vinyl-buffer');
//var sourcemaps = require('gulp-sourcemaps'); // Uncomment to use sourcemaps
var glob = require('glob');
var es = require('event-stream');
var rename = require('gulp-rename');
var footer = require('gulp-footer');
var gutil = require('gulp-util');
var gsize = require('gulp-filesize');
var debowerify = require('debowerify');

var CONFIG = require('../tasks/config');

gulp.task('browserify', function () {
  glob('./'+CONFIG.SOURCE_ROOT+'/**/*.main.{js,coffee,jsx}',
function (err, files) {
  var tasks = files.map(function (entry) {
    return browserify({ entries: [entry], debug: true })
      .transform(coffeeify)
      .transform(reactify)
      .transform(debowerify)
      .bundle()
      .pipe(source(entry))
      .pipe(rename(function (filepath) {
        filepath.basename = filepath.basename.replace("-main", "");
        filepath.dirname = "";
        filepath.extname = ".bundle.js";
      }));
    /* Uncomment the next two lines for client-side-source maps for
debugging */
    // .pipe(sourcemaps.init({loadMaps: true}))
    // .pipe(sourcemaps.write('./'))
    .pipe(footer(CONFIG.FOOTER_TEXT))
    .pipe(gsize())
    .pipe(gulp.dest('./public/javascripts'))
    .on('error', gutil.log);
  });
  es.merge(tasks);
});
});

return gulp.src('*');
});
```

6.13.4 tasks/generate-config.js

```
/* Include the necessary modules */
var gulp = require('gulp');
var fs = require('fs');

/* If configuration files don't exist, create them
(they'll still need populating with keys and credentials) */
gulp.task('generate-config', function () {
  fs.open('config/src/keys.coffee', 'r', function (err) {
    if (err && err.code === 'ENOENT') {
      fs.createReadStream('config/src/sample-keys.coffee')
        .pipe(fs.createWriteStream('config/src/keys.coffee'));
    }
  });
});
```

6.13.5 tasks/scripts.js

```
var gulp    = require('gulp');
var gsize   = require('gulp-filesize');
var jshint   = require('gulp-jshint');
//var uglify = require('gulp-uglify'); // Uncomment to minify (see below aswell)
var coffee  = require('gulp-coffee');
var coflint = require('gulp-coffeelint');
var footer  = require('gulp-footer');
var filter  = require('gulp-filter');
var argv    = require('yargs').argv;
var gulpIf  = require('gulp-if');

var CONFIG  = require('../tasks/config');
var devMode = argv.dev ? true : CONFIG.SHOW_OUTPUT;

/* JavaScript Tasks */
gulp.task('scripts', function(){

    /* A list of file sources and destinations */
    var paths = CONFIG.SCRIPT_PATHS;

    /* For each file path, run all script tasks */
    var tasks = [];
    paths.forEach(function(path) {
        tasks.push(awsesomeizeScripts(path.src, path.dest));
    });

    return (tasks); // Return all results as array of multiple streams

    /* Function applies all the tasks on the script files and returns a pipe dest
    */
    function awsesomeizeScripts(srcPath, resPath){

        /* Filters to be applied (so that different operations can be done on
        different files) */
        var bundleFilter = filter(['*', '!**/*-main.{js,coffee}', '!**/*-
main.{js,coffee}']);
        var coffeeFilter = filter('**/*.coffee', {restore: true}); // MUST be
declared here in order to RESET correctly!

        return gulp.src(srcPath)
            .pipe(bundleFilter) // Ignore browserify files

            /* CoffeeScript tasks */
            .pipe(coffeeFilter)
                .pipe(coflint())
                .pipe(gulpIf(devMode, coflint.reporter()))
                .pipe(coffee())
            .pipe(coffeeFilter.restore)
            .pipe(jshint())
            .pipe(gulpIf(devMode, jshint.reporter('jshint-stylish')))
            //.pipe(uglify()) // Uncomment line to minify output JavaScript
            .pipe(footer(CONFIG.FOOTER_TEXT))
            .pipe(gulpIf(devMode, gsize()))
            .pipe(gulp.dest(resPath));
    }
});
```

6.13.6 tasks/styles.js

```
/* Include the necessary modules */
var gulp    = require('gulp');
var gutil   = require('gulp-util');
var gsize   = require('gulp-filesize');
var concat  = require('gulp-concat');
var less    = require('gulp-less');
var minCss  = require('gulp-minify-css');
//var cssLint = require('gulp-csslint'); // Uncomment to lint CSS
var changed = require('gulp-changed');
var footer  = require('gulp-footer');
var es      = require('event-stream');
var gulpIf  = require('gulp-if');
var argv    = require('yargs').argv;

var CONFIG  = require('../tasks/config');

var devMode = argv.dev ? true : CONFIG.SHOW_OUTPUT;

/* CSS and Less Tasks */
gulp.task('styles', function(){
  var cssSrcPath = CONFIG.SOURCE_ROOT + '/' + CONFIG.CSS_SRC_DIR_NAME;
  var cssResPath = CONFIG.DEST_ROOT + '/' + CONFIG.CSS_DEST_DIR_NAME;

  var cssFromLess = gulp.src([cssSrcPath+'/*.less'])
    //.pipe(recess())
    //.pipe(recess.reporter())
    .pipe(less());
  var cssFromVanilla = gulp.src([cssSrcPath+'/*.css']);

  var excludedCss = gulp.src(cssSrcPath+'/*/*.css');

  var excludedLess = gulp.src(cssSrcPath+'/*/*.less')
    //.pipe(recess())
    //.pipe(recess.reporter())
    .pipe(less());

  var concatenatedCss = es.merge(cssFromLess, cssFromVanilla)
    .pipe(concat(CONFIG.CSS_FILE_NAME));

  return es.merge(concatenatedCss, excludedCss, excludedLess)
    .pipe(changed(cssResPath))
    //.pipe(cssLint()) // Uncomment to lint CSS
    //.pipe(gulpIf(devMode, cssLint.reporter()))
    .pipe(minCss({compatibility: 'ie8'}))
    .pipe(gulpIf(devMode, gsize()))
    .pipe(footer(CONFIG.FOOTER_TEXT))
    .pipe(gulp.dest(cssResPath))
    .on('error', gutil.log);
});
```

6.13.7 tasks/test.js

```
var gulp    = require('gulp');
var mocha   = require('gulp-mocha');
var istanbul = require('gulp-istanbul');
var gulpIf  = require('gulp-if');
var argv    = require('yargs').argv;

var CONFIG = require('../tasks/config');
var devMode = argv.dev ? true : CONFIG.SHOW_OUTPUT;
var reportMode = devMode ? 'spec' : 'nyan';

require('coffee-script/register');

/* Run all tests and produce coverage report */
gulp.task('test', function (cb) {
  gulp.src(['client-side-source/**/*.js', 'main.js'])
    .pipe(istanbul()) // Covering files
    .pipe(istanbul.hookRequire()) // Force `require` to return covered files
    .on('finish', function () {
      gulp.src(['test/*.js,coffee'])
        .pipe(mocha({reporter: reportMode}))

        .pipe(gulpIf(devMode, istanbul.writeReports({dir:
'./reports/coverage-reports'})))
        .pipe(istanbul.enforceThresholds({ thresholds: { global: 90 } }));
    })
    .on('end', cb);
});

/* Run Mocha tests */
gulp.task('mocha', function () {
  return gulp.src('test/*.js,coffee')
    .pipe(mocha({reporter: 'list'}));
});
```

6.13.8 tasks/watch.js

```
var gulp    = require('gulp');
var CONFIG  = require('../tasks/config');

/* Configure files to watch for changes - NOT USED BY DEFAULT TASK */
gulp.task('watch', function () {

  /* For each CoffeeScript directory, watch, compile and reload */
  CONFIG.SCRIPT_PATHS.forEach(function (path) {
    gulp.watch(path.src, ['scripts']);
  });

  /* Watch build and re-bundle browserify files */
  gulp.watch(CONFIG.SOURCE_ROOT+'/**/*.-(main,module).{js,coffee}',
['browserify']);

  /* Watch compile and reload LESS, SASS and CSS */
  gulp.watch(CONFIG.SOURCE_ROOT+'/**/*.{css,less}', ['styles']);
});
```


6.13.9 tasks/nodemon.js

```
/* Include the necessary modules */
var gulp = require('gulp');
var nodemon = require('gulp-nodemon');
var bSync = require('browser-sync');

/* Nodemon task for monitor for changes with live restarting */
gulp.task('nodemon', function (cb) {
  var called = false;
  return nodemon({
    script: './bin/www',
    watch: ['client-side-source/**/*']
  })
  .on('start', function onStart() {
    if (!called) { cb(); }
    called = true;
  })
  .on('restart', function onRestart() {
    setTimeout(function reload() {
      bSync.reload({
        stream: false
      });
    }, 500);
  });
});
```

6.13.10 tasks/quick-coffeescript.js

```
var gulp = require('gulp');
var coffee = require('gulp-coffee');
var CONFIG = require('../tasks/config');

gulp.task('quick-coffeescript-watch', function() {
  gulp.watch('./**/*.coffee', ['quick-coffeescript']);
});

gulp.task('quick-coffeescript', function() {

  /* A list of file sources and destinations */
  var paths = CONFIG.SCRIPT_PATHS;

  /* For each file path, run all script tasks */
  var tasks = [];
  paths.forEach(function(path) { tasks.push(coffeeify(path.src, path.dest)); });
  return (tasks); // Return all results as array of multiple streams

  /* Function applies all the tasks on the script files & returns a pipe dest */
  function coffeeify(srcPath, resPath) {
    return gulp.src(srcPath).pipe(coffee()).pipe(gulp.dest(resPath));
  }
});
```

6.13.11 tasks/images.js

```
/* Include the necessary modules */
var gulp = require('gulp');

/* CSS and Less Tasks */
gulp.task('images', function() {
  return [
    gulp.src('client-side-source/gr/favicon.ico').pipe(gulp.dest('public')),
    gulp.src('client-side-s/gr/**/*.{png,gif,jpg}').pipe(gulp.dest(path))
  ];
});
```

6.13.12 tasks/clean.js

```
var gulp = require('gulp');
var del = require('del');

var scriptPaths = require('../tasks/config').SCRIPT_PATHS;

/* Delete built files */
gulp.task('clean', function (cb) {
  var paths = ['public/javascripts', 'public/stylesheets'];
  scriptPaths.map(function (sp) {
    paths.push(sp.dest+'/*.js');
  });
  del(paths).then(cb());
});
```

6.13.13 tasks/build.js

```
var gulp = require('gulp');
var runSequence = require('run-sequence');

/* Clean the work space */
gulp.task('build', function (cb) {
  runSequence('clean',
    ['scripts', 'browserify', 'styles', 'images'],
    cb);
});
```

6.13.14

6.13.15 tasks/config.js

```
/* Define constants */
var footerText = "\n/* (C) Alicia Sykes <alicia@aliciasykes.com> 2015      "
+
  "*/\r\n/* MIT License. Read full license at: https://goo.gl/IL4lQJ */\r\n";

module.exports = {
  SOURCE_ROOT      : "client-side-source", // Folder name for all js and css
  client-side-source
  DEST_ROOT        : "public",           // Folder name for the results root
  CSS_DEST_DIR_NAME : "stylesheets",     // Name of CSS directory
  CSS_SRC_DIR_NAME  : "styles",          // Name of CSS directory
  JS_FILE_NAME      : "all.min.js",      // Name of output JavaScript file
  CSS_FILE_NAME     : "all.min.css",     // Name of output CSS file
  FOOTER_TEXT       : footerText,        // Optional footer text for output files
  SCRIPT_PATHS      : [
    { src: 'client-side-source/scripts/**/*.{js,coffee}',
      dest: 'public/javascripts' },
    { src: 'client-side-source/scripts/visualisations/*.coffee',
      dest: 'public/javascripts/charts' },
    { src: 'server-side-source/models/*.coffee',
      dest: 'models' },
    { src: 'server-side-source/utils/*.coffee',
      dest: 'utils' },
    { src: 'server-side-source/config/*.coffee',
      dest: 'config' },
    { src: 'server-side-source/routes/*.coffee',
      dest: 'routes' },
    { src: 'server-side-source/api-routes/*.coffee',
      dest: 'routes' }
  ],
  SHOW_OUTPUT: true
};
```

6.14 MODULE 1 – SENTIMENT ANALYSIS

6.14.1 index.coffee

```
afinnWordList = require __dirname + '/AFINN-111.json' # Get the AFINN-111 list

# Returns a boolean true if given word is found in word list
doesWordExist = (word)->
  if word of finnnWordList then true else false

# Returns an integer value + or - sentiment score for given word
getScoreOfWord = (word)->
  if finnnWordList[word] then finnnWordList[word] else 0

# Formats sentence and returns a lowercase a-z array of words
getWordsInSentence = (sentence)->
  sentence = if sentence? then sentence else '' # Double check is defined
  sentence = if typeof sentence == 'string' then sentence else ''
  sentence = sentence.toLowerCase()
  sentence = sentence.replace(/(?:https?|ftp):\/\/[^\s]+/g, '') # Remove URLs
  sentence = sentence.replace(/^[^\w\s]/gi, '') # Remove special characters
  sentence = sentence.split(' ') # Split into an array
  sentence = sentence.filter((n) -> n != '') # Remove blanks
  sentence = removeDuplicates(sentence)

# Remove Duplicates
removeDuplicates = (arr) ->
  if arr.length == 0
    return []
  res = {}
  res[arr[key]] = arr[key] for key in [0..arr.length-1]
  value for key, value of res

# Ensure score is in a valid range between -1 to +1
scaleScore = (score)->
  score = if score > 10 then 10 else score
  score = if score < -10 then -10 else score
  score/10

# Returns an overall sentiment score for sentence
analyseSentence = (sentence) ->
  score = 0
  wordsArr = getWordsInSentence(sentence)
  for word in wordsArr
    if doesWordExist(word)
      score += getScoreOfWord(word)
  scaleScore(score)

module.exports = analyseSentence # Export main method as module

# If we're developing/ testing then export the private methods too
if process.env.NODE_ENV == 'test'
  module.exports =
    main: analyseSentence
    _private:
      scaleScore: scaleScore
      doesWordExist: doesWordExist
      getScoreOfWord: getScoreOfWord
      removeDuplicates: removeDuplicates
      getWordsInSentence: getWordsInSentence
```

6.14.2 .gitignore

```
node_modules
.idea
*.log
reports
dev.js
coverage
```

6.14.3 .travis.yml

```
language: node_js
node_js:
  - "0.12"
  - "0.10"

before_script:
  - "npm i -g mocha"
```

6.14.4 dev.js

```
/**
 * FOR DEVELOPMENT ONLY.
 * NOT TO BE PUSHED TO GIT!
 */

var sentimentAnalysis = require('./index');

console.log(sentimentAnalysis('Dinosaurs are awesome!'));
console.log(sentimentAnalysis('Everything is stupid!'));
console.log(sentimentAnalysis('Windows is very unstable!'));
console.log(sentimentAnalysis('The slug was tired, he felt sluggish!'));
console.log(sentimentAnalysis('London is gloomy today because of all the smog!'));
console.log(sentimentAnalysis('I am so grateful for all the presents, thanks!'));
console.log(sentimentAnalysis('Really enjoying the warm weather!'));
console.log(sentimentAnalysis('It was a catastrophic disaster!'));
```

6.14.5 index.js (compiled)

```
(function(){var e,r,n,t,o,i,u;e=require(__dirname+"/AFINN-111.json"),n=function(r){return r in e?!0:!1},t=function(r){return e[r]?e[r]:0},o=function(e){return e=null!=e?e:""},e="string"===typeof e?e:"",e=e.toLowerCase(),e=e.replace(/(?:https?|ftp):\/\/[\\n\\S]+/g,""),e=e.replace(/^[\\w\\s]/gi,""),e=e.split(""),e=e.filter(function(e){return""!==e}),e=i(e),i=function(e){var r,n,t,o,i,u;if(0===e.length)return[];for(o={},n=r=0,t=e.length-1;t>=0?t>=r:r>=t;n=t>0?++r:--r)o[e[n]]=e[n];i=[];for(n in o)u=o[n],i.push(u);return i},u=function(e){return e=e>10?10:e,e=-10>e?-10:e,e/10},r=function(e){var r,i,s,c,f;for(s=0,f=o(e),r=0,i=f.length;i>r;r++)c=f[r],n(c)&&(s+=t(c));return u(s)},module.exports=r,"test"===process.env.NODE_ENV&&(module.export s={main:r,_private:{scaleScore:u,doesWordExist:n,getScoreOfWord:t,removeDuplicates:i,getWordsInSentence:o}})}.call(this);
```

6.14.6 package.json

```
{
  "name": "sentiment-analysis",
  "version": "0.1.1",
  "description": "Sentiment analysis module using AFINN-111",
  "main": "index.js",
  "scripts": {
    "test": "gulp test",
    "cover": "istanbul cover node_modules/mocha/bin/_mocha"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/Lissy93/sentiment-analysis.git"
  },
  "keywords": ["sentiment", "analysis", "sentiment", "analysis", "twitter",
    "AFINN", "AFINN-111", "emotion", "detection", "valence"
  ],
  "author": "Alicia Sykes <sykes.alicia@gmail.com> (http://aliciasykes.com)",
  "license": "MIT",
  "bugs": {
    "url": "https://github.com/Lissy93/sentiment-analysis/issues"
  },
  "homepage": "https://github.com/Lissy93/sentiment-analysis#readme",
  "devDependencies": {
    "chai": "^3.3.0",
    "coffee-script": "^1.10.0",
    "del": "^2.0.2",
    "gulp": "^3.9.0",
    "gulp-coffee": "^2.3.1",
    "gulp-coffeelint": "^0.5.0",
    "gulp-footer": "^1.0.5",
    "gulp-istanbul": "^0.10.1",
    "gulp-mocha": "^2.1.3",
    "gulp-watch": "^4.3.5",
    "mocha": "^2.3.3"
  }
}
```

6.14.7 gulpfile.js

```
/**
 * The gulp file configures the gulp build setup
 * Run "gulp build" to built project and compile CoffeeScript
 * Run "gulp test" to run Mocha unit tests and Istanbul coverage tests
 * Run "gulp" to clean, build, test and watch for changes during development
 */

/* Require the node modules */
var gulp    = require('gulp');
var coffee  = require('gulp-coffee');
var lint    = require('gulp-coffeelint');
var del     = require('del');
var footer  = require('gulp-footer');
var mocha   = require('gulp-mocha');
var istanbul= require('gulp-istanbul');
var watch   = require('gulp-watch');
require('coffee-script/register');

var footerTxt = "\\/* (C) Alicia Sykes <alicia@aliciasykes.com> 2015      " +
  "\\*\\r\\n\\* MIT License. Read full license at: https://goo.gl/IL4lQJ *\\/*";

/* Delete the files currently in finished directory */
gulp.task('clean', function () {
  return del([ './index.js' ]);
});

/* Lint, compile and minify CoffeeScript */
gulp.task('build', ['clean'], function(){
  return gulp.src('./*.coffee')
    .pipe(lint())
    .pipe(lint.reporter())
    .pipe(coffee())
    .pipe(footer(footerTxt))
    .pipe(gulp.dest('./'));
});

/* Run unit tests and generate coverage report */
gulp.task('test', function (cb) {
  gulp.src(['./index.js'])
    .pipe(istanbul())
    .pipe(istanbul.hookRequire())
    .on('finish', function () {
      gulp.src('./test/**/*.coffee', {read: false})
        .pipe(mocha({ reporter: 'spec' }))
        .pipe(istanbul.writeReports({reporters: ['text-summary', 'lcov']}))
        //.pipe(istanbul.enforceThresholds({ thresholds: { global: 90 } }))
        .on('end', cb);
    });
});

/* Watch for changes and refresh */
gulp.task('watch', function(){
  gulp.watch('./**/*.coffee', ['test-after-build']);
  gulp.watch('./test/**/*.coffee', ['test']);
});

/* Don't test, until it has built, to avoid file not found errors */
gulp.task('test-after-build', ['build'], function(){
  gulp.start('test')
});

/* Default gulp task, deletes old files, compiles source files and runs tests */
gulp.task('default', ['test-after-build', 'watch']);
```

6.14.9 readme.md

```
# sentiment-analysis
[![Build Status](https://travis-ci.org/Lissy93/sentiment-
analysis.svg?branch=dev)](https://travis-ci.org/Lissy93/sentiment-analysis)
[![Dependency Status](https://david-dm.org/lissy93/sentiment-
analysis.svg)](https://david-dm.org/lissy93/sentiment-analysis)
[![devDependency Status](https://david-dm.org/lissy93/sentiment-analysis/dev-
status.svg)](https://david-dm.org/lissy93/sentiment-analysis#info=devDependencies)
[![Code Climate](https://codeclimate.com/github/Lissy93/sentiment-
analysis/badges/gpa.svg)](https://codeclimate.com/github/Lissy93/sentiment-
analysis)
> [AFINN-111] based Sentiment analysis module

## Install
`npm install sentiment-analysis --save`

## Usage
```javascript
var sentimentAnalysis = require('sentiment-analysis');

sentimentAnalysis('Dinosaurs are awesome!'); // +0.4
sentimentAnalysis('Everything is stupid!'); // -0.2
sentimentAnalysis('Windows is very unstable!'); // -0.2
sentimentAnalysis('London is gloomy today because of all the smog!'); // -0.4
sentimentAnalysis('I am so grateful for all the presents, thank you!'); // +0.5
sentimentAnalysis('Really enjoying the warm weather!'); // +0.3
sentimentAnalysis('It was a catastrophic disaster!'); // -0.6
```

sentiment-analysis will return a score between -1 and +1, where negative numbers
represent a negative overall sentiment.

## Testing
`npm test`

See unit test, integration testing results on [Travis CI]

## Development
See the `gulpfile.js` for documentation of build process.

## License
MIT  [Alicia Sykes](http://aliciasykes.com)

[AFINN-111]: <http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010>
[Travis CI]: <https://travis-ci.org/Lissy93/sentiment-analysis>
```

6.14.10 test/mocha.opts

```
--compilers coffee:coffee-script/register
--reporter spec
```

6.14.11

6.14.12 test/utills.test.coffee

```
expect = require('chai').expect
process.env.NODE_ENV = 'test'
sentimentAnalysis = require('../index')._private
```

```
describe 'doesWordExist will return boolean weather word exists', ()->
  doesWordExist = sentimentAnalysis.doesWordExist

  it 'should return a boolean value', ()->
    expect(doesWordExist('coffee')).to.be.a('boolean')
    expect(doesWordExist('mocha')).to.be.a('boolean')
    expect(doesWordExist('java')).to.be.a('boolean')

  it 'should return true for words that exist', () ->
    expect(doesWordExist('woo')).to.be.true
    expect(doesWordExist('alive')).to.be.true
    expect(doesWordExist('awesome')).to.be.true
    expect(doesWordExist('anger')).to.be.true
    expect(doesWordExist('bright')).to.be.true
    expect(doesWordExist('love')).to.be.true
    expect(doesWordExist('easy')).to.be.true
    expect(doesWordExist('drunk')).to.be.true
    expect(doesWordExist('dumb')).to.be.true
    expect(doesWordExist('hacked')).to.be.true
    expect(doesWordExist('important')).to.be.true
    expect(doesWordExist('hug')).to.be.true
    expect(doesWordExist('itchy')).to.be.true
    expect(doesWordExist('laugh')).to.be.true
    expect(doesWordExist('stupid')).to.be.true
    expect(doesWordExist('bomb')).to.be.true

  it 'should return false for words that do not exist', () ->
    expect(doesWordExist('hello')).to.be.false
    expect(doesWordExist('world')).to.be.false
    expect(doesWordExist('everything')).to.be.false
    expect(doesWordExist('is')).to.be.false
    expect(doesWordExist('stupidness')).to.be.false
    expect(doesWordExist('acid')).to.be.false
    expect(doesWordExist('dinosaurs')).to.be.false
    expect(doesWordExist('laptop')).to.be.false
    expect(doesWordExist('pepsi')).to.be.false
    expect(doesWordExist('lorem')).to.be.false
    expect(doesWordExist('ipsum')).to.be.false
    expect(doesWordExist('squashed')).to.be.false
    expect(doesWordExist('watson')).to.be.false
    expect(doesWordExist('brain')).to.be.false

  it 'should not throw an error with funny values', ()->
    expect(doesWordExist(1)).to.be.a('boolean')
    expect(doesWordExist([])).to.be.a('boolean')
    expect(doesWordExist(true)).to.be.a('boolean')
    expect(doesWordExist(undefined)).to.be.a('boolean')
    expect(doesWordExist(1)).to.be.false
    expect(doesWordExist([])).to.be.false
    expect(doesWordExist(undefined)).to.be.false

describe 'getScoreOfWord method return a sentiment score for that word', ()->
  getScoreOfWord = sentimentAnalysis.getScoreOfWord
```



```

it 'should return an integer', ()->
  expect(getScoreOfWord('amazing')).to.be.a('number')
  expect(getScoreOfWord('warm')).to.be.a('number')
  expect(getScoreOfWord('yummy')).to.be.a('number')

it 'should be in a range of -5 to + 5', () ->
  expect(getScoreOfWord('nice')).to.be.above(-5).to.be.below(5)
  expect(getScoreOfWord('good')).to.be.below(5).to.be.below(5)
  expect(getScoreOfWord('great')).to.be.above(-5).to.be.below(5)
  expect(getScoreOfWord('awesome')).to.be.above(-5).to.be.below(5)

it 'should return 0 if word doesn\'t exist, rather than crashing', ()->
  expect(getScoreOfWord('batman')).equal(0)
  expect(getScoreOfWord('superman')).equal(0)
  expect(getScoreOfWord('spiderman')).equal(0)
  expect(getScoreOfWord('pepperpig')).equal(0)

it 'should return 0 if passed multiple words at a time that don\'t exist', ()->
  expect(getScoreOfWord('type error')).equal(0)
  expect(getScoreOfWord('everything is stupid')).equal(0)
  expect(getScoreOfWord('dinosaurs are awesome')).equal(0)

it 'should return actual positive score for positive words that exist', ()->
  expect(getScoreOfWord('united')).equal(1)
  expect(getScoreOfWord('unstoppable')).equal(2)
  expect(getScoreOfWord('excited')).equal(3)
  expect(getScoreOfWord('win')).equal(4)
  expect(getScoreOfWord('outstanding')).equal(5)

it 'should return actual negative score for negative words that exist', ()->
  expect(getScoreOfWord('fight')).equal(-1)
  expect(getScoreOfWord('fails')).equal(-2)
  expect(getScoreOfWord('evil')).equal(-3)
  expect(getScoreOfWord('fraud')).equal(-4)
  expect(getScoreOfWord('twat')).equal(-5)

it 'should return 0 for neutral words that exist', ()->
  expect(getScoreOfWord('some kind')).equal(0)
  # There is only 1 neutral result in the AFINN word list!

describe 'getWordsInSentence will transform a sentence into a clean array', ()->
  getWordsInSentence = sentimentAnalysis.getWordsInSentence

  it 'Should correctly turn a sentence into an array', ()->
    expect(getWordsInSentence('hello world')).eql(['hello', 'world'])
    expect(getWordsInSentence('this is a longer sentence'))
      .eql(['this', 'is', 'a', 'longer', 'sentence'])

  it 'Should normalise case', ()->
    expect(getWordsInSentence('HeLlO wOrLd')).eql(['hello', 'world'])
    expect(getWordsInSentence('JAVASCRIPT')).eql(['javascript'])

  it 'Should remove duplicates', ()->
    expect(getWordsInSentence('foo foo bar foo'))
      .eql(['foo', 'bar'])
    expect(getWordsInSentence('foo foo BAR Foo bAr foO bar foo'))
      .eql(['foo', 'bar', ])

  it 'Should remove blanks', ()->
    expect(getWordsInSentence('space      blank      '))
      .eql(['space', 'blank'])

  it 'Should remove special characters', ()->
    expect(getWordsInSentence('foo ! ^&*^&^%&^^&%^bar$$$^'))
      .eql(['foo', 'bar'])

describe 'removeDuplicates should remove duplicates from an array', () ->
  removeDuplicates = sentimentAnalysis.removeDuplicates

```

```

it 'should remove duplicates', () =>
  expect(removeDuplicates(['hello', 'world', 'hello', 'hello']))
    .eql(['hello', 'world'])

describe 'scaleScore should ensure the score is within the valid range', () =>
  scaleScore = sentimentAnalysis.scaleScore

  it 'should not be below -1', () =>
    expect(scaleScore(-1.2)).to.be.above(-1.01)
    expect(scaleScore(-38.8)).to.be.above(-1.01)
    expect(scaleScore(1.2)).to.be.above(-1.01)

  it 'should not be above +1', () =>
    expect(scaleScore(4.5)).to.be.below(1.01)
    expect(scaleScore(42)).to.be.below(1.01)
    expect(scaleScore(-1.2)).to.be.below(1.01)

  it 'should have 1 or 2 decimal places', () =>
    expect(scaleScore(1)).to.be.within(-1,+1);
    expect(scaleScore(-1)).to.be.within(-1,+1);
    expect(scaleScore(0)).to.be.within(-1,+1);
    expect(scaleScore(10)).to.be.within(-1,+1);
    expect(scaleScore(-1)).to.be.within(-1,+1);
    expect(scaleScore(-1.01)).to.be.within(-1,+1);
    expect(scaleScore(+1.0001)).to.be.within(-1,+1);
    expect(scaleScore(999999)).to.be.within(-1,+1);
    expect(scaleScore(-999999)).to.be.within(-1,+1);
    expect(scaleScore(-0)).to.be.within(-1,+1);
    expect(scaleScore(+0)).to.be.within(-1,+1);
    expect(scaleScore(42)).to.be.within(-1,+1);
    expect(scaleScore(3.1415926535897932)).to.be.within(-1,+1);
    expect(scaleScore(-273.15)).to.be.within(-1,+1);

```

6.14.13

6.14.14 test/main.test.coffee

```

expect = require('chai').expect
process.env.NODE_ENV = 'test'
sentimentAnalysis = require('../index').main

describe 'Check the modules basic functionality', () =>
  it 'should return an integer', () =>
    expect(sentimentAnalysis('lorem ipsum dolor seit amet'))
      .to.be.a('number')
    expect(sentimentAnalysis('foo bar')).to.not.be.undefined;

  it 'Should return the correct sentiment value for negative sentences', () =>
    expect(sentimentAnalysis('I hate everything, everything is stupid')).equal(-0.5)
    expect(sentimentAnalysis('London is gloomy today because of all the smog')).equal(-0.4)
    expect(sentimentAnalysis('He was captured and put into slavery')).equal(-0.3)
    expect(sentimentAnalysis('Windows is very unstable')).equal(-0.2)
    expect(sentimentAnalysis('The slug was tired, he felt sluggish')).equal(-0.2)

  it 'Should return the correct sentiment value for positive sentences', () =>
    expect(sentimentAnalysis('Today is a wonderful amazing awesome day')).equal(1)
    expect(sentimentAnalysis('I am so grateful for all the presents, thank you!')).equal(0.5)

  it 'Should not return a score greater than 1 of smaller than -1', () =>
    expect(sentimentAnalysis('happy happy amazing awesome cool'))
      .to.be.above(-1.1).to.be.below(1.1)
    expect(sentimentAnalysis('crap crap crap crap'))
      .to.be.above(-1.1).to.be.below(1.1)

```

6.15 MODULE 2- FETCH TWEETS

```
request = require 'request'
querystring = require 'querystring'

class FetchTweets

  shouldFormatResults = null

  constructor: (@credentials, @shouldFormatResults = true) ->
    shouldFormatResults = @shouldFormatResults

  # Fetches the data from given URL from Twitter
  makeRequest = (url, credentials, callback) ->
    oauth = {
      callback: '/'
      consumer_key: credentials.consumer_key
      consumer_secret: credentials.consumer_secret
    }
    request {
      url: url
      json: true
      oauth: oauth
    }, (error, response, body) ->
      if !error and response.statusCode == 200
        callback (body)

  # Processes the results to get rid of not needed data
  formatResults = (twitterResults) ->
    if !shouldFormatResults
      return twitterResults
    tweetObjescts = []
    for rawTweetObject in twitterResults.statuses
      tweetObjescts.push({
        'date': rawTweetObject.created_at,
        'body': rawTweetObject.text
        'location': prepareLocation(rawTweetObject)
        'retweet-count' : rawTweetObject.retweet_count
        'favorited-count' : rawTweetObject.favorite_count
        'lang' : rawTweetObject.lang
      })
    tweetObjescts

  # Get location
  prepareLocation = (body) ->
    location =
      place_name: '_'
      location: { lat: 0.0000000, lng: 0.0000000 }

  # Check Coordinates object
  if body.coordinates?
    location.location.lat = body.coordinates.coordinates[1]
    location.location.lng = body.coordinates.coordinates[0]
  else if body.geo?
    location.location.lat = body.geo.coordinates[0]
    location.location.lng = body.geo.coordinates[1]

  # Check for place name
  if body.place?
    location.place_name = body.place.name
  else if body.user?
    location.place_name = body.user.location
```

```

location

formatTrends = (preResults) ->
  results = []
  for item in preResults[0].trends
    results.push {trend: item.name, volume: item.tweet_volume}
  results

byTopic: (params = '', cb) ->

  if typeof params is 'string'
    urlParams = 'q='+params+'&count=100'
  else if typeof params is 'object'
    urlParams = querystring.stringify(params)

  url = 'https://api.twitter.com/1.1/search/tweets.json?' + urlParams
  makeRequest url, @credentials, (results) ->
    cb formatResults(results)

trending: (placeId, cb) ->
  trendingUrl = 'https://api.twitter.com/1.1/trends/place.json'+ '?id='+placeId
  makeRequest trendingUrl, @credentials, (results) -> cb formatTrends results

closestTrendingWoeid: (lat, long, cb) ->
  url="https://api.twitter.com/1.1/trends/closest.json?lat=#{lat}&long=#{long}"
  makeRequest url, @credentials, (results) -> cb results

module.exports = FetchTweets

```

6.16 MODULE 3 – STREAM TWEETS

```
request = require 'request'
querystring = require 'querystring'

_private = {}

class StreamTweets

  shouldFormatResults = null # Will be set (true|false) when object initiated

  constructor: (@credentials, @shouldFormatResults = true) ->
    shouldFormatResults = @shouldFormatResults

  # Fetches the data from given URL from Twitter
  makeRequest = (params, credentials, callback) ->

    message = "" # Variable that accumulates tweets
    separator = "\r" # What separates multiple Tweet objects

    params.oauth = {
      callback: '/'
      consumer_key: credentials.consumer_key
      consumer_secret: credentials.consumer_secret
      token : credentials.token
      token_secret : credentials.token_secret
    }

    req = request.post(params, (err) ->
      console.error err if err
    )

    req.on 'data', (buffer) ->
      message += buffer.toString()
      tweetSeparatorIndex = message.indexOf(separator)
      didFindTweet = tweetSeparatorIndex != -1

      if didFindTweet
        tweet = message.slice(0, tweetSeparatorIndex)
        if isValidJson(tweet)
          callback JSON.parse(tweet)
          message = message.slice(tweetSeparatorIndex + 1)

    req.on 'disconnect', (reason) -> console.log reason

  # Processes the results to get rid of not needed data
  formatResults = (twitterResults) ->

    if !shouldFormatResults
      return twitterResults
    {
      'date': twitterResults.created_at,
      'body': twitterResults.text
      'location': prepareLocation(twitterResults)
      'retweet-count' : twitterResults.retweet_count
      'favorited-count' : twitterResults.favorite_count
      'lang' : twitterResults.lang
    }

  # Get location
  prepareLocation = (body) ->
    location =
      place_name: '_'
      location: { lat: 0.000000, lng: 0.000000 }

  # Check Coordinates object
```

```

if body.coordinates?
  location.location.lat = body.coordinates.coordinates[1]
  location.location.lng = body.coordinates.coordinates[0]
else if body.geo?
  location.location.lat = body.geo.coordinates[0]
  location.location.lng = body.geo.coordinates[1]

# Check for place name
if body.place?
  location.place_name = body.place.name
else if body.user?
  location.place_name = body.user.location

location

# Check if the string would be valid json
isStrValidJson = (str) ->
  try JSON.parse str
  catch e then return false
  true

# Public function, to be directly called by main program
stream: (params, cb) ->
  #Check what type of params we working with, and format appropriately
  if typeof params is 'string' then urlParams = 'track='+params
  else if typeof params is 'object' then urlParams =
    querystring.stringify(params)

  params = {
    uri: 'https://stream.twitter.com/1.1/statuses/filter.json?'+urlParams
  }
  makeRequest params, @credentials, (results) ->
    cb formatResults(results)

  _private = {
    isStrValidJson: isStrValidJson
    formatResults: formatResults
    makeRequest: makeRequest
  }

module.exports = StreamTweets

# If we're developing/ testing then export the private methods too
if process.env.NODE_ENV == 'test'
  module.exports = {
    main: StreamTweets
    _private: _private
  }

```

6.17 MODULE 4 – REMOVE WORDS

```
fs = require('fs')

_private = {}

removeWords = (sentence, getRidOfDuplicates=true, wordsArray=undefined) ->
  if !wordsArray?
    wordsArray = fs.readFileSync(__dirname + '/words.txt', 'utf8').split('\r\n')
  else
    wordsArray =
      if typeof wordsArray == 'string' then [wordsArray] else wordsArray
    wordsArray = formatWordsArr(wordsArray)

  sentence = if typeof sentence == 'string' then sentence else ''
  sentenceArr = arrayifySentence(sentence)

  for dictionaryWord in wordsArray
    for sentenceWord, index in sentenceArr
      if sentenceWord == dictionaryWord
        sentenceArr.splice(index,1)
  if getRidOfDuplicates then return removeDuplicates(sentenceArr)
  else return sentenceArr

# Format the sentence and return an array
arrayifySentence = (sentence) ->
  sentence = formatSentence(sentence) # Lowercase + remove special chars
  sentence = sentence.split(' ') # Split into an array
  sentence = sentence.filter((n) -> n != '') # Remove blanks

# Formats each element of the words array
formatWordsArr = (wordsArr) ->
  if !wordsArr instanceof Array
    return []
  for word, i in wordsArr
    wordsArr[i] = formatSentence(word)
  wordsArr

# Removes URLs, special characters and then de-capitalizes a string
formatSentence = (sentence) ->
  sentence = if sentence? then sentence else '' # Double check is defined
  sentence = sentence.toLowerCase()
  sentence = sentence.replace(/(?:https?|ftp):\/\/[^\s]+\/g, '') # Remove URLs
  sentence = sentence.replace(/[^\w\s]/gi, '') # Remove special characters

removeDuplicates = (arr) ->
  if arr.length == 0
    return []
  res = {}
  res[arr[key]] = arr[key] for key in [0..arr.length-1]
  value for key, value of res

_private = {
  arrayifySentence: arrayifySentence
  formatWordsArr: formatWordsArr
  formatSentence: formatSentence
  removeDuplicates: removeDuplicates
}

module.exports = removeWords

# If we're developing/ testing then export the private methods too
if process.env.NODE_ENV == 'test'
  module.exports = {
    main: removeWords
    _private: _private
  }
```

6.18 MODULE 5 - PLACE LOOKUP

```
request = require 'request'
querystring = require 'querystring'

host = "https://maps.googleapis.com/maps/api/place/textsearch/json"

removeStingChars = (str)->
  str.replace(/[^\A-Za-z0-9\s,]/g, '')

makeURL = (paramaters, apiKey)->
  urlParams =
    if typeof paramaters is 'string' then 'query='+ removeStingChars paramaters
    else if typeof paramaters is 'object' then querystring.stringify(paramaters)
  host + '?' + 'key=' + apiKey + '&' + urlParams

formatResults = (body)->
  place_name: body.results[0]['formatted_address']
  location: body.results[0]['geometry']['location']

makeRequest = (url, requestCallback)->
  request { url: url, json: true },
  (error, response, body) ->
    if !error and response.statusCode == 200 then requestCallback(body)

main = (paramaters, apiKey, callback)->
  url = makeURL(paramaters, apiKey)
  makeRequest(url, (results)->
    try callback(formatResults(results))
    catch e then callback({error: 'Zero results returned'})
  )

module.exports = main
```


6.19 MODULE 6 – TWEET LOCATION

```
request = require 'request'

main = (placeID, credentials, callback, formatResults = false)->
  url = "https://api.twitter.com/1.1/geo/id/#{placeID}.json" # The endpoint
  oauthCredentials = { # The keys
    consumer_key : credentials.consumer_key,
    consumer_secret : credentials.consumer_secret,
    token: credentials.token,
    token_secret: credentials.token_secret
  }

  # Make the request
  request {
    url: url
    json: true
    oauth: oauthCredentials
  }, (error, response, body)->
    if !error and response.statusCode == 200
      callback(if formatResults then stripDown(body) else body)

# Optionally get rid of all pointless data and just return lat and lon for GMaps
stripDown = (body) ->
  if body.centroid?
    lat: Math.round(body.centroid[1] * Math.pow(10, 6)) / Math.pow(10, 6),
    lon: Math.round(body.centroid[0] * Math.pow(10, 6)) / Math.pow(10, 6)
  else
    {}
module.exports = main
```

6.20 MODULE 7 – HP HAVEN SA

```
request = require 'request'
querystring = require 'querystring'

main = (paramaters, apiKey, callback)->
  host = "https://api.havenondemand.com/1/api/sync/analyzesentiment/v1"

  if typeof paramaters is 'string'
    urlParams = 'text='+paramaters
  else if typeof paramaters is 'object'
    urlParams = querystring.stringify(paramaters)

  url = host + '?' + urlParams + '&apikey=' + apiKey

  request {
    url: url
    json: true
  }, (error, response, body) ->
    if !error and response.statusCode == 200
      callback(body)

module.exports = main
```

6.21 MODULE 8 – HP HAVEN EXTRACT ENTITIES

```
request = require 'request'
querystring = require 'querystring'

entities = ['people_eng', 'places_eng', 'companies_eng', 'organizations',
  'languages', 'drugs_eng', 'professions', 'universities',
  'films', 'internet', 'teams', ]

makeUrl = (paramaters, apiKey) ->
  host = "https://api.havenondemand.com/1/api/sync/extractentities/v2"

  if typeof paramaters is 'string'
    urlParams = 'text='+paramaters
    for entity in entities then urlParams += '&entity_type='+entity
    urlParams += '&show_alternatives=false'
  else if typeof paramaters is 'object'
    entities = paramaters.entity_type
    urlParams = querystring.stringify(paramaters)

  url = host + '?' + urlParams + '&apikey=' + apiKey

formatResults = (body) ->
  results = {}
  for b in body.entities
    if !results[b.type] then results[b.type] = []
    matches = []
    for m in b.matches then matches.push m.original_text
    additionalInformation = {}
    additionalInformation.wiki =
      if b.additional_information.hasOwnProperty 'wikipedia_eng'
        b.additional_information.wikipedia_eng
      else ''

    additionalInformation.image =
      if b.additional_information.hasOwnProperty 'image'
        b.additional_information.image
      else ''

    results[b.type].push({
      normalized_text: b.normalized_text
      matches: matches
      additional_information: additionalInformation
    })
  results

module.exports = (paramaters, apiKey, callback)->
  url = makeUrl paramaters, apiKey # Make the URL

  # Make the actual request, and call the callback
  request {url: url, json: true}, (error, response, body) ->
    if !error and response.statusCode == 200 then callback formatResults body
    else callback(error)
```

6.21.1

6.22 MODULE 9 – FIND REGION FROM LOCATION

```
fs = require('fs')

_private = {}

# Takes the raw CSV string and returns a nice list of JSON region objects
convertCsvToJson = (csvRegions) ->
  regions = []
  for r in csvRegions.splice 1, csvRegions.length # For each line in CSV file
    r = r.match(/(".*?"|^[^",\s]+)(?=\s*|\s*$)/g) # Break into array
    for e, i in r then r[i] = r[i].replace(/["]+/g, '').trim() # Neaten
    regions.push { # Create a JSON object for region, and push to results
      country: r[0], alpha2_code: r[1], alpha3_code: r[2],
      numeric_code: r[3], latitude: Number(r[4]), longitude: Number(r[5])
    }
  regions # Done, return regions

# Returns a region object closest to a given latitude and longitude
findRegion = (lat, lng) ->

  # Inject regions from CSV file
  csvRegions = fs.readFileSync(__dirname + '/regions.csv', 'utf8').split('\r\n')

  # Convert regions to JSON
  regions = convertCsvToJson csvRegions

  # Find difference between our location and region location
  for r in regions
    r.diff = Math.round(Math.abs(r.latitude - lat) + Math.abs(r.longitude - lng))

  # Sort regions by closest first
  regions.sort (a, b) -> parseFloat(a.diff) - parseFloat(b.diff)

  regions[0] # Return closest region

# Create and export functions
module.exports.country = (lat, lng) -> findRegion(lat, lng).country
module.exports.alpha2_code = (lat, lng) -> findRegion(lat, lng).alpha2_code
module.exports.alpha3_code = (lat, lng) -> findRegion(lat, lng).alpha3_code
module.exports.numeric_code = (lat, lng) -> findRegion(lat, lng).numeric_code
module.exports.regionObject = findRegion
```

7 APPENDIX SEVEN

TECH STACK

This appendix document briefly lists all the external packages, libraries, frameworks and opensource languages that were utilised to create the final solution

Twitter Sentiment Visualisations

The research and development of a sentiment analysis module, and the implementation of it on real-time social media data, to generate a series of live visual representations of sentiment towards a specific topic or by location in order to find trends.

Alicia Sykes 12011471

Oxford Brookes University



Library 1 D3.js



Language 1 Node.js



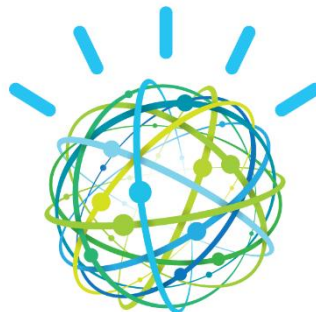
Database 1 MongoDB



Framework 1 Express.js



Language 2 CoffeeScript



API 1 IBM Watson



Language 3 Jade



Language 7 JavaScript



Language 6 HTML5



Language 5 CSS3



Language 4 Less



Framework 2 Socket.io



Language 8 Markdown



Library 2 jQuery



Dev Tools 7 GitHub



Dev Tools 5 Gulp



API 3 Google Places

UNDERSCORE.JS

Library 3 Underscore.js



Dev Tools 8 JetBrains WebStorm



API 2 HP Haven OnDemand



Dev Tools 6 Require.js



Dev Tools 4 Bower



Dev Tools 2 Node Package Manager



Dev Tools 9 Code Climate



Dev Tools 1 Chai and Mocha

