MSc Data Science
Department of Computer Science and Information Systems
Birkbeck, University of London, 2019

Project Proposal

Twitter Sentiment Analysis: Big Data Software Application Development

Samuel McIlroy

This proposal is substantially the result of my own work, expressed in my own words, except where explicitly indicated in the text. I give my permission for it to be submitted to the JISC Plagiarism Detection Service.

Abstract

The objective of the project is the real-time sentiment analysis of Twitter data. In this proposal I describe the development of a data intensive software application to acquire, store and process large volumes of Twitter tweet data generated daily across London. This data will then be used to implement a sentiment analysis model at scale. From this, I will create a front end interface to produce maps and key visualisations of the general sentiment levels and trends across London areas and landmarks. The development will focus on three key areas: 1) The design and implementation of scalable data pipelines through an investigation into distributed computing and cloud technologies for dealing with data at scale. 2) The development and implementation of the sentiment analysis model. 3) The design and implementation of a web frontend interface to display the results.

Contents

1. Introduction

The overall objective of this project is to show insights into the general ‘happiness’ levels of Londoners. I will accomplish this through a sentiment analysis of tweets originating from different areas across the city and a mapping of the results and key trends. In order to accomplish this I will document the design and implementation of ‘big data’ technologies to acquire, process and store the tweets generated across London every second as close to real-time as possible. I will look at modern developments in open source technologies in distributed and cloud computing around the processing of large volumes of data efficiently which have seen rapid progress and widespread use over the course of the last decade.

It is estimated that the amount of data generated during this year, 2019, will surpass all data that has been collected in the last 500 decades combined [1]. Each day the internet search and social media giants alone collect and store petabytes of user generated content, as approximately 3.7 billion active internet users log in around the world [2].

The need to derive meaning and use cases from this vastly expanding data has led to rapid developments in distributed and cloud computing technologies to handle data processing at increasing scale. The principal challenge has been that as large volume data storage becomes cheaper and easier, relatively slow pace in CPU speed improvements has meant that the volume of data becomes the bottleneck, rather than the clock speed of the CPU in more traditional single machine development, if the data is to be used in a meaningful way and processed in a reasonable time-frame. The need to distribute data processing over many machines for parallel processing then becomes key, leading to the term ‘data-intensive’ software applications [3].

While just some years ago data processing at this scale would have been expensive and difficult due to the number of machines required and the relatively niche software engineering skill-sets required, modern advances in the cloud computing Infrastructure as a Service (IaaS) approach of providers (e.g. Amazon Web Services, Google Cloud Platform and Microsoft Azure) allowing cheap and easy access to rented computing clusters, and the development of, and industry preference for, free and open source (FOSS) software platforms for distributed computing and big data analytics (Apache Spark etc.) access to hundreds of machines, large scale storage and industry level software is available cheaply and easily to large companies, start ups and single end users alike. [3]

1.1 Problem Statement

I propose to obtain streaming real-time Twitter data from the Twitter API and complete an end to end investigation and implementation of three key stages:

1. Investigation, analysis and implementation of data pipelines to prepare and store the data required at scale.

2. Investigation and implementation of sentiment analysis techniques and solutions. Implementation of a chosen sentiment analysis model across the data at scale. Present evidence of testing and results.

3. Investigation and implementation of an appropriate web frontend to display maps and visualisations, as close to real time as Twitter API restrictions allow.

I have chosen this project as it will allow me to simulate the end to end development of a data software product which includes the deployment of a machine learning (sentiment analysis) problem at scale that would be developed by software engineers supporting data analytics teams in the real world. I intend to conduct documentation and implementation with reference to industry standards, practices and preferences.

Using Twitter data to map sentiment across London areas gives both access to the required amount of ‘big’ data to complete a software application in this area, and the opportunity to solve an interesting problem. Providing analysis and access to sentiment and opinion, especially when broken down geographically, allows us to quantify human feelings regarding their environments, situations and their daily lives. Access to such compiled and visualised data is of interest to data scientists, social scientists, corporations and marketing organisations, urban planners and more. [7]

Even by eye it is a difficult problem to solve quantifying sentiment in human language, and at any large scale becomes impractical, but using modern NLP techniques, especially at a large scale with the help of distributed computing and cloud technologies, we can start to gain meaningful insights. [9]

1.2 Aims and Objectives

I am aiming to develop and document my progress, decisions and final outputs through the three key stages of the project. My overall objective is to demonstrate the development of an industry standard software application to provide access to visualisations and results from large scale data processing and to show the practical application of a machine learning model at scale. The development and write up should be akin to the level expected of a software/data engineer in a large data science team.

The aims and objectives of the project in brief:

- The investigation, detailed analysis, and practical implementation of distributed and cloud technologies and large scale database and storage solutions to develop and document, as close to industry standard as possible, the data pipelines required for both streaming and batch processing of data from the Twitter API as needed for the development of the subsequent visualisation frontend. I will provide outlines of the technologies both considered and used, appropriate documentation and write up of all code written.

- The development of the sentiment analysis model, details of appropriate existing software libraries, and documentation of any tweaking and theory applied to the model to improve results. A write up of the process, documentation of results obtained, the application of the model at scale will be be provided along with any code produced.

- The development of a frontend visualisation/mapping of the sentiment analysis being performed on the streaming data. Further develop the front end to include other interesting visualisations/displays. Document the choice of frontend technology, initial designs and the implementation and code write up. The frontend application should be tested and then made available throughout the marking period for examiners to access and use.

1.3 Challenges, Volume of Data and Proof of Concept Approach

As also discussed in the subsequent background chapter, there are challenges to the proposed analysis of sentiment across all London twitter users. The principal challenge is the volume of data that can be acquired from the Twitter API. In general, the Twitter API limits the amount of tweets that can be captured to a 1% sampling of the data. This means that the tweets I can collect to analyse do not represent the entire population.

To estimate the volume of data I can expect, I have conducted a one hour test of streaming data using my own computer and a simple connection to the API through the Tweepy python library, which is detailed further in the next chapter. During this test I was able to capture 1753 tweets in a one hour period. Assuming a constant rate during the day, for simplicity, I can estimate to process approximately 42,000 tweets per day, or around one tweet every two seconds.

This volume of data should give sufficient volume to implement the technologies I outline in this proposal and to provide an adequate sampling of data to create analysis and mappings which are representative of the population as a whole. While it would be possible to use alternative more traditional approaches to handle this data, I will focus on the implementation of a ‘proof of concept’ approach where if the volume of data were to be increased, to the full Twitter population for example, then the same approach could be taken with as little alterations as possible.

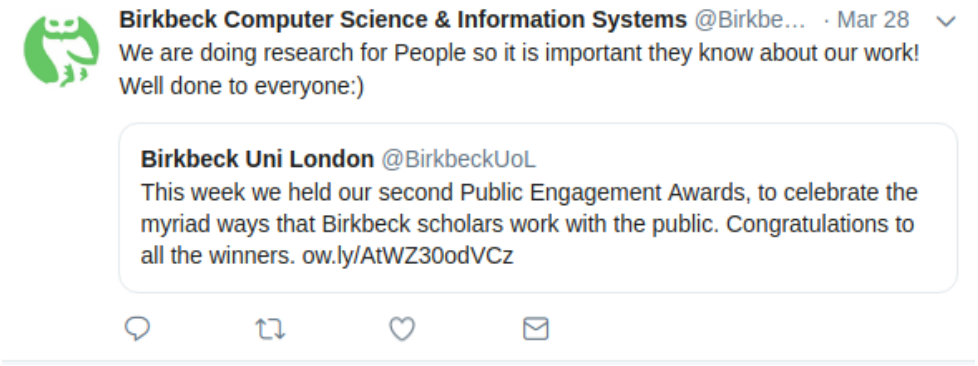
2. Background

This chapter aims to give a brief overview of the technologies and techniques discussed and considered for the application development in other chapters.

2.1 Twitter

Twitter [4] is an online social networking platform where users can post short, primarily text based although photos and gif attachments are becoming more common, messages of up to 240 characters, with the standard length of a message being approximately 33 characters [5]. Each message is known as a ‘tweet’ and Twitter generates, stores and processes over 400,000 tweets per minute containing text and photo based personal status updates from their over 300 million active users [2].

Users range from the general public to celebrities, public and political figures and accounts representing entities such as corporations, organisations, universities etc. Users can interact with each other by following a user, mentioning another user (for example “hello @Birkbeck_CS) or by highlighting an important topic through adding hashtags (short statements beginning #) to their messages (e.g. #DataScience). Users can also share (‘retweet’) another user to their own feed and followers, ‘like’ a comment to show support, or directly comment a tweet of their own to another.



A typical tweet, highlighting a response to another account

The Twitter API

Twitter provide a standard Streaming API for developers. This will allow me to collect tweets as they are posted. In order to access the APIs users must apply and be accepted as a developer. I have applied and have access to the APIs.

Twitter currently limits the standard Streaming API to a 1% sample of the streaming tweets queried. To navigate this restriction, some aspects of the project may have to use samples or more limited data but should still allow the aims to be implemented, at least as proofs of concept which could be bettered with less restrictive access. Premium APIs are available to remove some of these restrictions but these are very likely to be cost prohibitive to implement in this project. [38]

Geotags

As part of the data available from the Twitter API, if a user has tweeted from a specific location a geotag giving that location as close as possible is provided. A geotag is essentially a coordinate which can be located on a map. [8]

Using the geotags attached to the tweet data I have successfully tested obtaining tweets generated by users in London by providing a bounding box (four geotags which form a square area from which tweets should be sourced):

	Country	Place	Coordinates	User	Date	Tweet
0	United Kingdom	South East - England	[[[-1.957296 50.574606] [-1.957296 52.196285] [1.451788 52.196285] [1.451788 50.574606]]]	@Liamz1998	2019-04-14 17:22:19	I LOVE YOU LIVERPOOL!
1	United Kingdom	Islington - London	[[[-0.142058 51.518552] [-0.142058 51.575301] [-0.076305 51.575301] [-0.076305 51.518552]]]	@IwonaWho	2019-04-14 17:22:24	@AWildDanRS And sorry next time I will share in the chat here as well
2	United Kingdom	Hammersmith - London	[[[-0.254563 51.463873] [-0.254563 51.532901] [-0.177653 51.532901] [-0.177653 51.463873]]]	@Margybargy	2019-04-14 17:22:24	@Chrest_brett @KetoCarnivore @xeyedmess @ProfTimNoakes Of course. What can concern us is that much science quoted b… https://t.co/2wLIRam7as
3	Reino Unido	Inglaterra - Reino Unido	[[[-6.365194 49.882531] [-6.365194 55.811649] [1.768926 55.811649] [1.768926 49.882531]]]	@mariamg2006	2019-04-14 17:22:28	#Msec_Tel_STEM https://t.co/ED1WMriWbD
4	United Kingdom	Camberwell - London	[[[-0.111476 51.419425] [-0.111476 51.509947] [-0.029731 51.509947] [-0.029731 51.419425]]]	@peckham65till	2019-04-14 17:22:29	@BrexitCentral I'm had enough of the Tories Labour and the CHUK ups I'm sick of the MSM I'm had enough of… https://t.co/N2y3M59sTd

Tweets can then be filtered by area and by specific coordinates. This will be the key to sampling the correct tweets for analysis.

2.2 Sentiment Analysis

Sentiment analysis is a natural language processing technique which aims to extract intended opinion from natural language text data. Different areas of sentiment analysis are tasked with understanding different aspects of language and will be investigated during the project to determine appropriate techniques to create the most effective results.

Polarity Classification

Polarity classification is the modeling of sentiment analysis as a machine learning classification problem where large volumes of unstructured text data, like those of feeds from the Twitter API, attempt to quantify overall intent in the language used by assigning a positive or negative classification. [9]

For example, polarity classification could assign each tweet in the data with a discrete 1 or a -1 value for positive and negative classification. This could be used to analyse the general average sentiment in each London area.

Fine Grained Sentiment Analysis

Fine grained sentiment analysis techniques aim to classify the data into more than a binary positive or negative. For example, quantifying the sentiment of the text on a sliding scale from very positive to very negative. Using a fine grained as opposed to binary classification would allow for more realistic results at the cost of higher difficulty and accuracy concerns. [9]

This would for example expand on the above polarity classification by providing continuous sentiment values between -1 and 1. This would allow for more accurate and in depth heat maps of each area and allow for more nuances in the language used in tweets.

Emotion Detection

Rather than polarity, a positive or negative classification only, emotion detection is a more advanced technique which leans heavily on linguistics and nuances of language to determine specific emotions: anger, sadness, happiness, which can be deduced form the text in addition to the overall polarity. [9]

Sentiment Analysis at Scale

Sentiment analysis techniques and programming libraries are increasingly impressive in their accuracy and ease of use. However the more data that can be put into the system, the better the results can be expected to be and the ability to deploy models at scale is key in creating useful results. [9] It has often be seen that improvements in the volume of the data that can be used and the speed at which the data can be processed have more impact than fine tuning of the models themselves. In 2001 Banko and Bill, in their paper ‘Scaling to Very Very Large Corpora for Natural Language Disambiguation’, noted that across many different NLP classification problems more data resulted in higher accuracy across the board. [10][11]

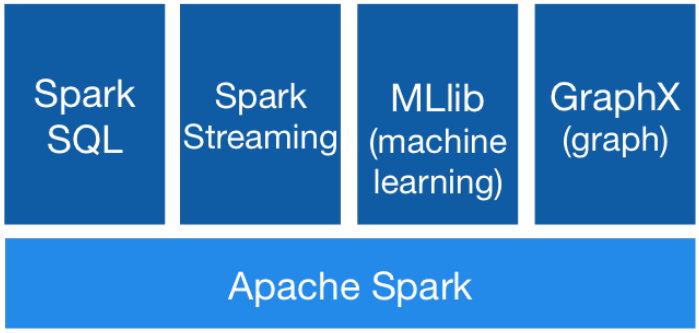
Challenges

As with a human reading of a written opinion, sentiment analysis techniques can struggle with ambiguities in the language such as double meanings, sarcasm, intent, exaggeration etc. which can cloud the intention of the opinion's source. [9]

2.3 Data Processing

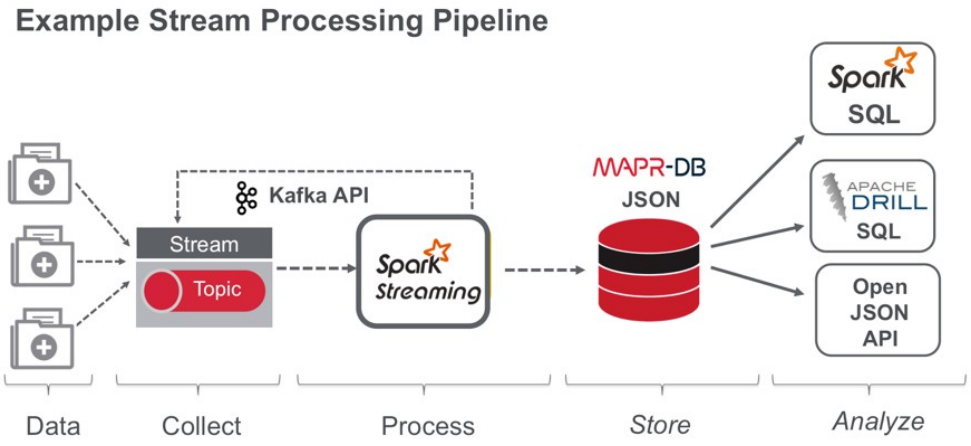
Apache Spark

Apache Spark is a technology and set of programming libraries designed for parallel data processing on multiple computers and is becoming a standard tool for big data processing.



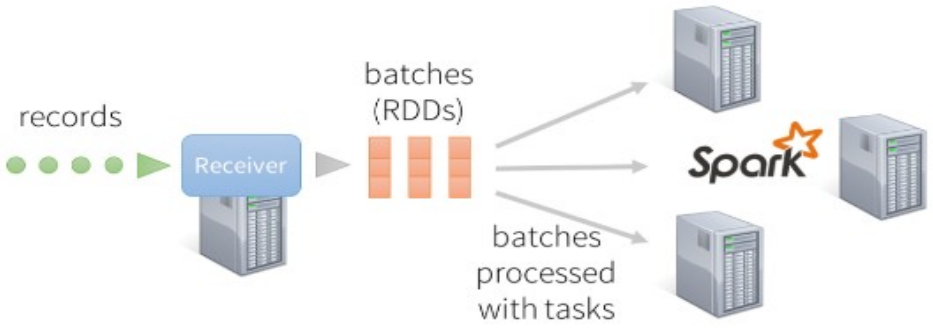
Apache Spark Ecosystem

Spark is incredibly fast due to its focus on in-memory processing of the data, rather than relying on data stored in stable storage as with comparable platforms such as MapReduce. Therefore Spark is particularly suited, due to its speed, for real time processing/streaming where data must be processed quickly to prevent bottlenecks as new data becomes available. [39]

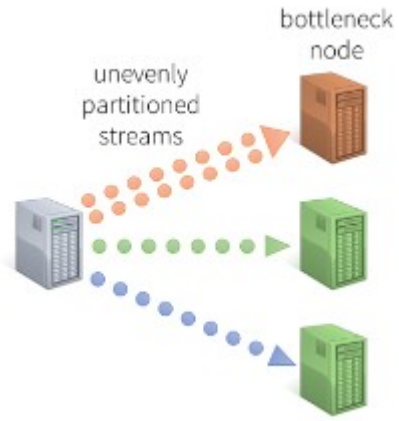


Apache Spark filling the data processing role in a typical streaming process [48]

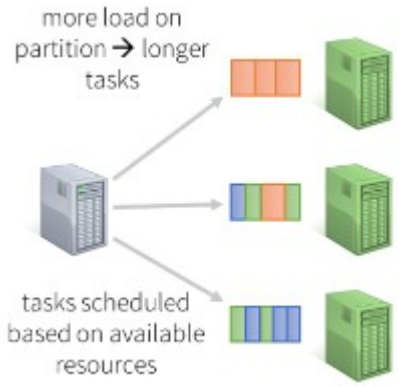
When Apache Spark receives data from a stream, rather than processing the records individually, the data is split into micro batches known as Resilient Distributed Dataframes which each contain milliseconds worth of the stream. These are then then processed as tasks in parallel within multiple computing nodes in a cluster: [47]



Processing the data in this way, as many tiny tasks, means that there is a great deal of flexibility in the way tasks are allocated to resources. In a traditional distributed data approach a longer task can cause bottlenecks:



whereas with many small tasks, longer jobs can have smaller tasks assigned to available resources:



This is all run in-memory as much as possible, although the data will be moved to disk storage for processing if resources are taken. This concept of scheduling smaller tasks across the nodes is known as Dynamic Load Balancing. [47]

MLLib

MLlib is a library of Apache Spark for machine learning, especially at scale in production. MLlib provides the ability to produce the required sentiment analysis for the project including the training and testing of the model at scale. [39]

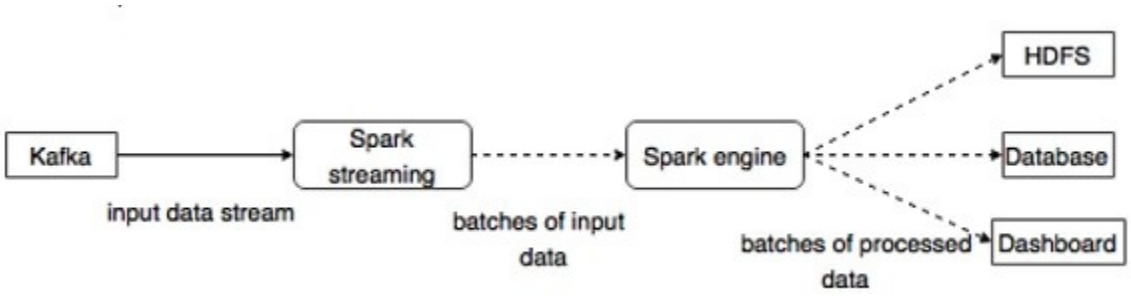
SparkSQL

SparkSQL is a tool built in to Apache Spark which allows data processed to be queried using standard SQL syntax. This will be useful in looking at overall features and trends in the data for some of the additional visualisations in addition to the sentiment analysis mapping. [39]

2.4 Message Brokers

In terms of the data streamed from the Twitter API, each tweet and its accompanying metadata can be seen as a message, or event. A message broker, such as Apache Kafka is designed to sit between the data processing stage and Twitter API. The broker can be set to listen for specific messages, such as a tweet with the corresponding London coordinates, and pull and store that message for later processing.

For example, in a typical streaming data pipeline, Apache Kafka might serve as the message broker to store data for and serve data to Apache Spark for processing:



Apache Kafka serving as a message broker, feeding streaming input data to Spark Streaming [49]

A message broker can be beneficial in that records can be stored there until it is picked up for processing, into Apache Spark for example. For instance, if you wanted to make changes or tweaks to the data processing then the message broker could continue to run and collect the data. If data was simply ‘pulled’ into the data processing as objects then if changes were required the pipeline would have to be shut down, and any data/tweets that were generated by the API during that time would be lost. [40]

Apache Kafka

Apache Kafka was developed at LinkedIn as a message broker designed to connect their internal systems as they moved to a distributed computing approach. Kafka is often seen as an appropriate choice for big data pipelines as it integrates well with the approach and fits within the popular Apache ecosystem. [18] [20]

RabbitMQ

RabbitMQ is seen as a common alternative to Apache Kafka and performs a complimentary function. RabbitMQ does have some disadvantages to consider for my use case, including a lack of Python support. [18] [20]

2.5 Database Storage

NoSQL Databases

NoSQL databases, unlike traditional Relational Database Management Systems, do not require a structure on the data and do not rely on traditional tables to stores the data. You do not have to prepare the schema for the data you expect beforehand.

This makes them an ideal solution for working with large volumes of unstructured data (data that does not have a clear structure such as tweets, server logs, sensor data etc.) as commonly seen in streaming applications. NoSQL databases are also known for their high transaction speed as they do not enforce traditional ACID (Atomicity, Consistency, Isolation, Durability) rules. [50]

Hbase

Apache Hbase is a an open source NoSQL database with capabilities for working with data in the scale of billions of rows. It is integrated with Apache Hadoop and would provide a good solution for storing the large volumes of unstructured Twitter data. Hbase is well known for its use in heavy text analysis based applications which would make it an ideal choice for this project. [18] [26]

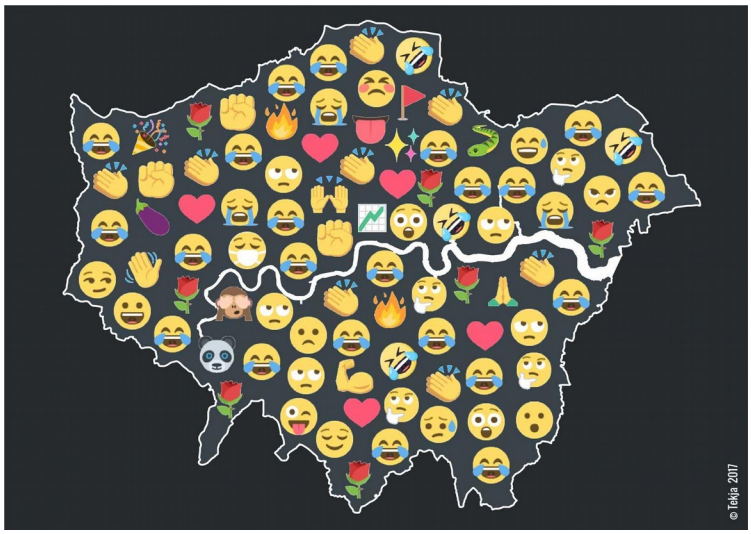
Apache Cassandra

Apache Cassandra is an alternative NoSQL database to consider with similar capabilities to Apache Hbase. [27] Cassandra users often prefer it for its in built query language, while Hbase offers better performance with fast/streaming data reads. [28]

2.6 Existing Solutions and Inspiration

Museum of London ‘Pulse’ Installation

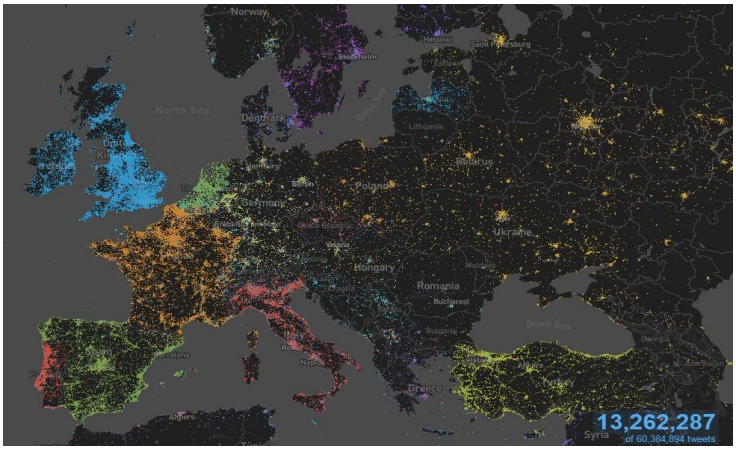
An exhibit produced by London data visualisation based studio Tekja for display in the Museum of London. Analysed twitter sentiment from Londoners, displaying sentiment, key figures and emoji usage on large electronic displays in the museum. [12]



Emoji Map of London

TweetMap

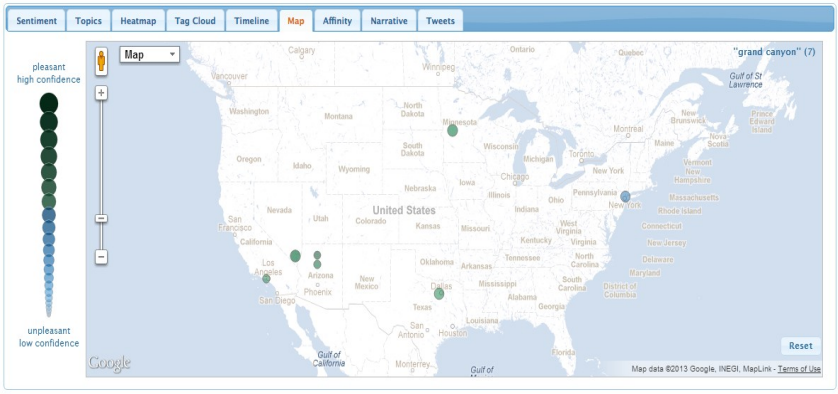
A live colour coded twitter map showing breakdowns of key information, searchable by world region, able to show individual tweets. Developed by OmniSci to demonstrate their GPU accelerated analytics capabilities. [13]



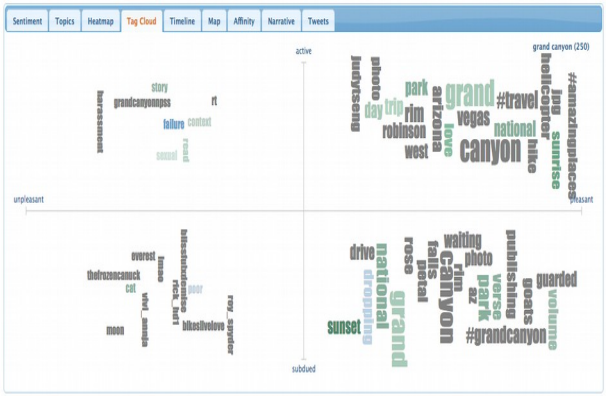
Tweet Heat Map by Country

Visualizing Twitter Sentiment, North Carolina State University

A similar project from North Carolina State University providing a web front end to visualise tweets and sentiment. [14]



Mapping



Word Clouds

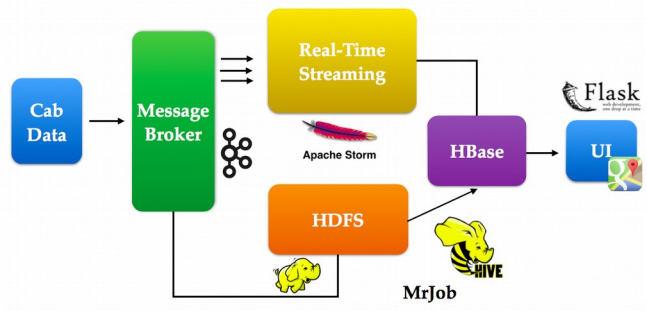
London Olympics

During the 2012 London Olympics, a team from the Massachusetts Institute of Technology attempted to light up the London eye in colours based on the sentiment of users tweeting about the games. [15]

London Eye Lit by Sentiment Levels

3. Development and Technologies

In the chapter I outline the development of the project stages and rationale for my initial choice of technologies. I will aim to develop and end to end process for the collection and presentation of the data.



A typical end to end data application pipeline/architecture [16]

3.1 Data Pipeline Development

In the first part of the project I will focus on the development of scalable and robust data pipelines to handle the acquisition, cleaning, structuring, processing and storage of large volumes of unstructured data fed from the Twitter API. I will be dealing with the sample streaming data from the API as the initial proof of concept but I will also keep in mind scaling to much larger volumes of data in future. With this there will be considerations to make at each stage of the pipeline development, both in regards to the challenges in working with the data and in selecting the most appropriate technologies.

3.1.1 Consideration Criteria

Working with very large volumes of data, especially unstructured data, brings unique challenges when moving from more traditional ways of working with smaller volumes and per-structured data sources.

The following are important criteria I will consider when designing the pipeline and choosing technologies which work well at each stage, and with each other.

Load

Load refers to the load placed on the system by the volume and velocity of the data which is being acquired and processed or to the number of active users. The higher the number of tweets collected per second for example, the more memory will be required and the more writes per second will be made to the database. High load can lead to bottlenecks which prevent consistent flow of the data through the pipeline. [3]

Fault Tolerance

Fault tolerance refers to the ability of, and extent to which, a system responds to faults, hardware failures, power outages, losses of virtual machines etc. A system which is flexible to fault tolerance is one in which data is processed in parallel and can allow a system to continue if individual nodes are compromised. [3]

Reliability

Reliable systems perform the required data processing under the required load with adequate fault tolerance and can be expected to do so on a continual basis. Reliable systems are also resistant to attack. [3]

Scalability

A scalable system performs reliably as load increases. When designing a pipeline should we consider only the current use case, can we expect more load in future etc.? An ideal scaling system in a distributed computing environment is one in which increasing the number of machines assigned to a task will negate issues with increased load. [3]

3.1.2 Pipeline Stages

There are various stages in a pipeline that should be considered in its design which I will detail here in the proposal. At each stage I have also very briefly outlined technologies to include in the comparative analysis in more depth which in the final project will form my chosen pipeline design and technology choices.

Stage One: Data Ingestion

The data ingestion stage involves connecting to and acquiring data from available data sources/APIs prior to conducting processing on the data to prepare it for analysis, running a machine learning model etc.

Important considerations at this stage are load and fault tolerance. [16]

Data ingestion will be handled by a message broker, as detailed in the background chapter. I have chosen Apache Kafka firstly for it's well documented Python support and for its suitability for the collection of fast streaming data. Kafka also has a strong reputation as a fault tolerant system that is easy for beginners to work with. [16]

Stage Two: Real-Time Processing/Streaming

Once data has been sourced and collected from its various sources it is ready to be processed. Real-time or 'stream' processing involves processing data from a source live as the data is collected. [17] For example a live feed of tweets made each minute available through the Twitter standard Streaming API.

I will be working with Apache Spark at this stage as it is widely seen as the standard tool in industry for streaming data processing. Documentation is excellent as is community support. Apache Spark also has strong Python support through it's Python implementation known as PySpark. [41]

Stage Three: Database Storage

The final stage in the pipeline design is the choice of database storage. Decision criteria at this stage are load, reliability and scalability. [16]

I will be storing the final results and data in long-term database storage, rather than only processing and utilising the data as it comes in/for a short time. The database will serve as the backend of the visualisation frontend and will also enable data to be analysed and stored over a long period of time thus enabling analysis and additional visualisations around trends in the data as time progresses.

At this stage I considered both Apache HBase and Apache Cassandra for their abilities to work with both structured and unstructured data and for my own interest in NoSQL database solutions. While both serve a similar function, HBase' faster read speeds make it more suitable in general for working with streaming data. [18][26][27][28]

Programming Languages

This project will be developed in the Python programming language. I am most experienced in Python programming and the majority of the open source Apache based technologies I have considered have robust Python implementations.

Apache Spark especially is known for its Python support and 'first class' treatment of Python as a second language. Spark was originally coded in Scala and there are documented speed advantages to using Scala over Python as Python requires additional steps to communicate with the JVM. However, these speed advantages are becoming less relevant as development and strong support for Python continues. [51]

There are also advantages to using the same language throughout the whole project as much a possible. Readability of code and documentation will be easier for example when using a single language.

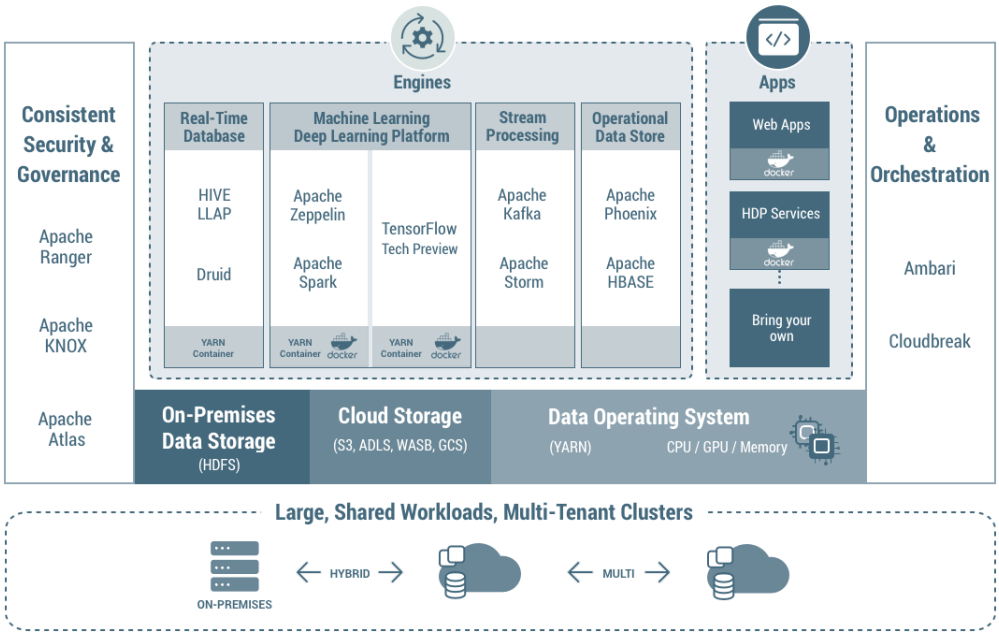
Development Environments

All of the technologies I have proposed are open source and can be installed and run on my own machine for initial development and testing. However, the dependencies between them, for example version requirements for Java and Python, must be carefully maintained. It can be a complex process to set up the required environments and maintain them during development.

Two enterprise software companies, Cloudera [29] and Hortonworks [30], supply development environments pre-packaged and configured to run and develop big data applications with the multiple open source Apache projects, data stores and machine learning platforms combined into a single environment. Working with one of their solutions would provide me with an easy to set up and maintain development environment for the project without having to set up individual technologies separately.

The Hortonworks Data Platform [30] and Hortonworks Data Flow [31] for example are open source solutions which would give a good choice of options for minimal set up time and cost. The Hortonworks solutions are also pre-available on cloud platforms including Microsoft Azure. [18]

Hortonworks Data Platform



Cloud Computing Providers

The development environments can be run on one machine locally for prototyping and development, however for the purposes of processing data at scale, especially the scales we would expect with a stream of the complete Twitter population, the application must be deployed either on a physical cluster of machines or on a cloud computing platform. Hardware restrictions, especially around physical memory requirements and the inability to run processing in parallel on a single machine, would limit the throughput of data and we would expect to see bottlenecks.

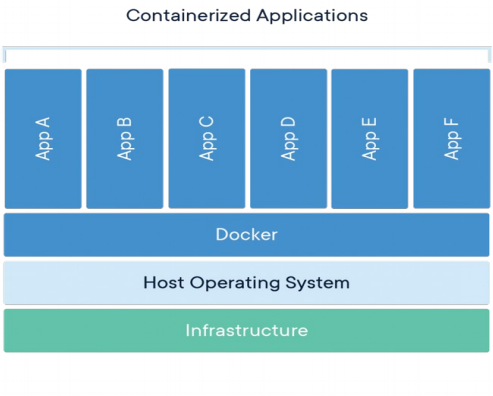
Running on a physical cluster would be restrictive in that the hardware required would be very expensive. Even if time on a University owned cluster could be arranged, it would be unlikely that, due to both the costs of running and needs of other staff and students, that the application could be allowed to run for significant amounts of time, limiting the amount of data being collected.

Cloud computing providers on the other hand provide for time on clusters of multiple machines and data storage servers to be rented The main providers to consider are Amazon Web Services [32], Google Cloud Platform [33] and Microsoft Azure [34]. All three also provide generous free academic credits/time for student and academic projects.

The final stage of the data pipeline development will be their successful deployment to a cloud platform. This is both the most cost effective solution and one which mirrors the mass exodus in industry from in house hardware to rented cloud computing resources.

Docker Containers

A Docker container is an executable software package that hosts together everything required for an application to run, including the OS, software tools, and dependencies. I am also considering using a Docker container to contain the application as a whole. The container could be created locally and then be deployed on any of the cloud providers above. [52]



A Docker Container [52]

3.2 Sentiment Analysis Model Development

Sentiment Analysis in Apache Spark MLlib

Sentiment analysis can be performed using the machine learning features of Apache Spark's MLlib and this is a popular and well documented method of applying sentiment analysis at scale. [18][36] MLlib supports the application of a range of machine learning classification models for sentiment analysis, including Naive Bayes Support Vector Machines (NB-SVMs) which is a hybrid algorithm with known strong performance in text sentiment classification. [42][43]

Once the model is achieving good results and accuracy on analysing the tweet data, through training and testing the model against pre-labelled data as outlined in the next chapter, I will then deploy the processing of the model into the deployed solution running on a chosen cloud platform.

3.3 Front End Visualisations Development

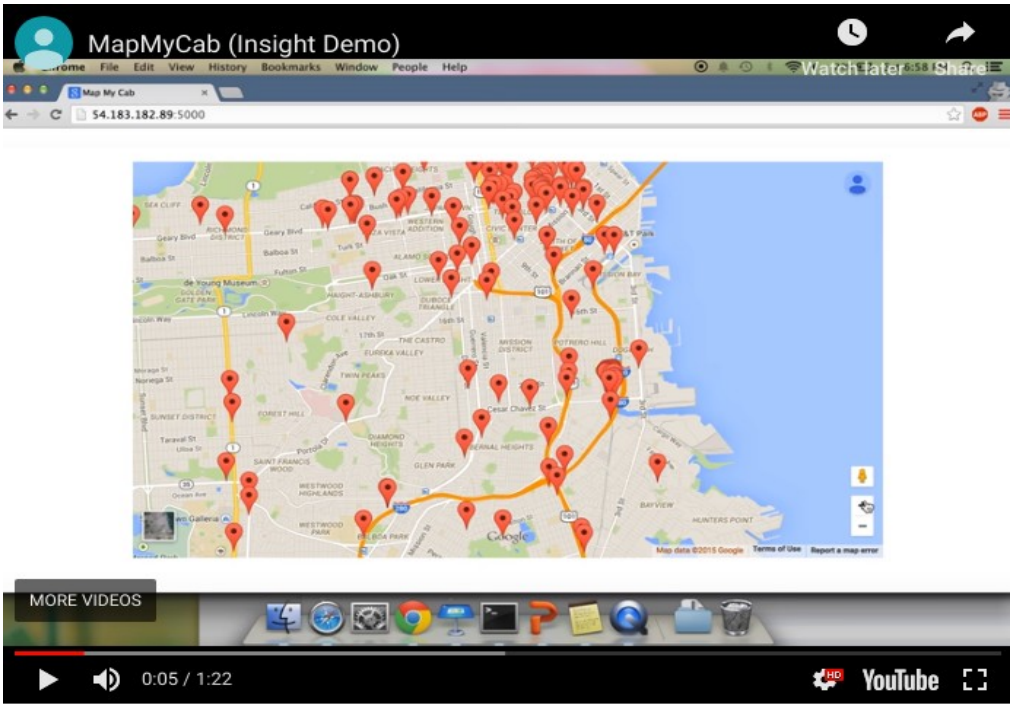
The frontend to the project will serve as the user interface and method of sharing results through interesting visualisations of the data. These are to include:

- A heatmap style map of London, showing sentiment levels in real-time and as trends over time (as data is continually analysed and collected in the backend database). This will be the main/showcase piece of the project.
- A heatmap style map of London showing heavy and light use of twitter in different areas.
- Word clouds showing common words and phrases used
- Popular hashtags, what are people talking about?
- Popular mentions, who are people talking about?

My primary driver in choosing a web framework for the frontend interface/visualisations was the need for something which is beginner friendly, due to my inexperience in web and frontend technologies, and for the framework to be based in Python, for my knowledge of the language and simplicity in developing in a single programming languages across each stage.

The two frameworks I considered for this stage are Flask [37] and Django [44]. Django is praised for web app development and is the most popular of the languages with a wider support community. On the other hand, Flask is seen as more lightweight. [45]

Due to my inexperience with web design and it's technicalities, I intend to produce initial visualisations in both to learn more about their design and implementation. I will document this process in the final project before selecting a single framework for the final product.

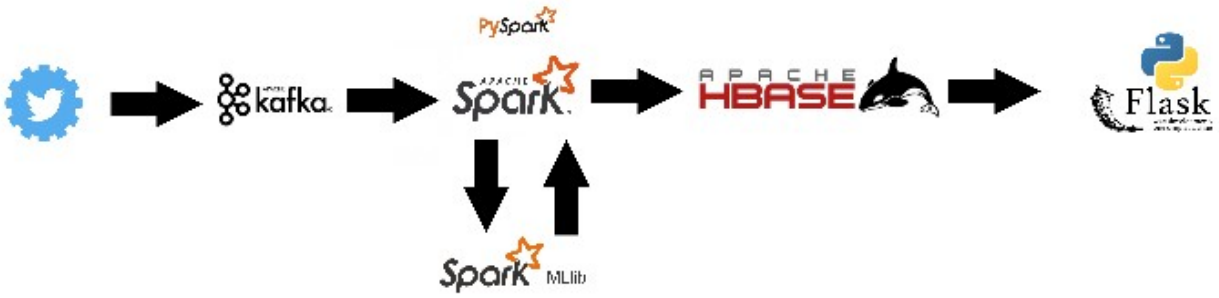


MapMyCab, an interactive API running in Flask based on San Francisco Cab Data [16]

3.4 Initial Outline

The following is an initial outline of the application to be developed, based on the discussion and comparison of technologies previously outlined in this proposal.

In general, Apache Kafka will serve as the data ingestion mechanism, listening for events (tweets in London) from the Twitter API. These will be collected and processed in Apache Spark with sentiment analysis through the Mllib library. Summaries of the data and results will be stored over time with HBase acting as a database and backend to the visualisations front end, which itself will be produced in Django/Flask:



This will serve as my starting off point for development. As I gain more experience with the tools and technologies any changes and justifications will be properly documented in the full project report.

4. Evaluation

The prototype system will be continually tested during the development process to ensure that the aims and objectives are being met.

Unit Testing

I will provide unit testing for all code developed to ensure that it is correct and running as expected. I will seek advice from my supervisor to ensure that unit testing is being done to correct standards and in the format expected for the languages being used.

Sentiment Model Training and Accuracy Testing

The sentiment model will be trained on archived data of tweets from previous years which are made available in full for analysis. The most common of these is Sentiment140, available through Kaggle [46] which provides 1,600,000 tweets with pre-labelled sentiment scores. A subset of the data will be used to train the data with the accuracy of the model tested against the remainder, matching the score generated by the model against the pre-labelled score.

Load Testing

The data pipelines will initially be tested working successfully against the expected load from the Twitter standard streaming API. Once this is running successfully the pipelines should be tested and examined under increasing load to simulate the effect of the introduction of larger samples of data should the application be applied to premium APIs.

This will be accomplished by using synthesised tweet data fed into the pipelines in addition to the real data. Synthesised data will be created using the above 1,600,000 sample tweets from the Sentiment140 dataset and assigning each tweet with a random geotag within London. [46]

Comparison with Similar Systems

Versions of the project will be compared with similar systems available online. I will compare available features and performance where available.

User Testing and Overall Feedback

Once initial versions of the project have been developed, I will seek user testing and feedback from colleagues and academics to create an iterative process where the final application is refined to a version ready for release/submission for marking. This will take take the form of a short questionnaire or set of user acceptance tasks. I will seek final overall feedback on a working version of the project from my supervisor before any version, with attached write up and documentation, is submitted.

5. Timeline

Week	Task
1	Investigation of data application and pipeline best practices
1	Setting up a development environment
1	Testing and prototyping data acquisition methods from the Twitter API
1	Cleaning and structuring of the data
2	Finalising and deploying data processing pipeline
4	Investigation of sentiment analysis techniques
4	Learning sentiment analysis in spark streaming applications
5	Development and testing of sentiment analysis model
5	Deploying the sentiment analysis mode
6	Design sentiment and other visualisations
7	Investigate use of Flask and Django frontend frameworks
9	Initial drafts of frontend
10	Complete frontend
11	User testing and feedback
13	Final Submission of Project

References

[1] Fassbender, Melissa, More data will be generated in 2019 than in the last 5,000 years, https://www.outsourcing-pharma.com/Article/2018/12/11/More-data-will-be-generated-in-2019-than-in-the-last-5-000-years, 2018

[2] Marr, Bernard, How Much Data Do We Create Every Day?, https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read 2018

[3] Kleppman, Martin, Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable and Maintainable Systems, O'Reilly Media, 2017

[4] Twitter, www.twitter.com

[5] Perez, Sarah, Twitter's Doubling of Character Count from 140 to 280 Had Little Impact on Length of Tweets, https://techcrunch.com/2018/10/30/twitters-doubling-of-character-count-from-140-to-280-had-little-impact-on-length-of-tweets, 2018

[6] Twitter Developers, https://developer.twitter.com

[7] Ballatore, Andrea, Birkbeck Bsc/MSc Research Themes, PS Mapping Place Sentiment in Social Media, https://docs.google.com/document/d/1YqGhZKp_8Fl_j2-PtYrVlaK-tiNelbgOj90Qt4huGmM/edit

[8] Twitter Developer Geo Guidelines, https://developer.twitter.com/en/developer-terms/geo-guidelines.html

[9] Sentiment Analysis: Nearly Everything You Need to Know, https://monkeylearn.com/sentiment-analysis

[10] Lin, Jimmy and Dyer, Chris, Data-Intensive Text Processing with Map Reduce, Morgan & Claypool Publishers, 2010

[11] Banko, Brill, Scaling to Very Very Large Corpora for Natural Language Disambiguation, 2001

[12] Museum of Lodon 'Pulse' Intstalltion, https://www.museumoflondon.org.uk/discover/pulse-capturing-londons-thoughts, 2017

[13] Tweet Map, https://www.omnisci.com/demos/tweetmap

[14] Healey & Ramaswamy, Visualizing Twitter Sentiment, North Carlolina State University, https://www.csc2.ncsu.edu/faculty/healey/tweet_viz, 2018

[15] Wasserman, Tom, Twitter Sentiment to Light Up London's Ferris Wheel, https://mashable.com/2012/07/24/londons-eye-twitter, 2012

[16] Kulshrestha, Preetika, How I Chose a Data Engineering Project, https://blog.insightdatascience.com/mapmycab-how-i-chose-a-data-engineering-project-75bd659c5eec, 2015

[17] Vaseekara, Gowthamy, Big Data Battle: Batch Processing vs Stream Processing, https://medium.com/@gowthamy/big-data-battle-batch-processing-vs-stream-processing-5d94600d8103, 2017

[18] Hortonworks, Building a Sentiment Analysis Application, https://hortonworks.com/tutorial/building-a-sentiment-analysis-application

[19] Hortonworks, What Apache NiFi Does, https://hortonworks.com/apache/nifi/#section_1

[20] Humphrey, Pieter, Understanding When to use RabbitMQ or Apache Kafka, https://content.pivotal.io/blog/understanding-when-to-use-rabbitmq-or-apache-kafka, 2017

[21] Hortonworks, Apache Hadoop, https://hortonworks.com/apache/hadoop/

[22] Hortonworks, Apache Spark, https://hortonworks.com/apache/spark/

[23] Hess, Ken, Hadoop vs. Spark: The New Age of Big Data, https://www.datamation.com/data-center/hadoop-vs.-spark-the-new-age-of-big-data.html, 2016

[24] Hortonworks, Apache Storm, https://hortonworks.com/apache/storm

[25] Educba, Apache Storm vs. Apache Spark – Learn 15 Useful Differences, https://www.educba.com/apache-storm-vs-apache-spark

[26] Hortonworks, Apache Hbase, https://hortonworks.com/apache/hbase

[27] Apache Cassandra, http://cassandra.apache.org

[28] Bekker, Alex, Cassandra vs. Hbase: Twins or Just Strangers with Similar Looks, https://www.scnsoft.com/blog/cassandra-vs-hbase, 2018

[29] Cloudera CDH, https://www.cloudera.com/developers/inside-cdh.html

[30] Hortonworks HDP, https://hortonworks.com/products/data-platforms/hdp

[31] Hortonworks HDF, https://hortonworks.com/tutorial/getting-started-with-hdf-sandbox

[32] Amazon Web Services, https://aws.amazon.com

[33] Google Cloud Platform, https://cloud.google.com

[34] Microsoft Azure, https://azure.microsoft.com

[35] Hortonworks, Deploying Hortonworks Sandbox on Microsoft Azure

[36] IBM, Real-time Sentiment Analysis of Twitter Hashtags with Spark, https://developer.ibm.com/clouddataservices/2016/01/15/real-time-sentiment-analysis-of-twitter-hashtags-with-spark, 2017

[37] Flask, http://flask.pocoo.org

[38] Adnan, Muhammed & Longley, Paul, Analysis of Twitter Usage in London, Paris, and New York City, University College London, 2013

[39] Zhang, Dell, Cloud Computing Week 11 Machine Learning in the Cloud, Birkbeck University of London, http://www.dcs.bbk.ac.uk/~dell/teaching/cc, 2019

[40] Kreps, Jay, It's Okay To Store Data In Apache Kafka, https://www.confluent.io/blog/okay-store-data-apache-kafka, 2017

[41] Apache Spark Python Pyspark Programming Guide, https://spark.apache.org/docs/0.9.0/python-programming-guide.html

[42] NBSVM-WEKA, https://vukbatanovic.github.io/project/nbsvm_weka

[43] Howard, Jeremy, Fast.ai Lesson 10, NLP, https://www.youtube.com/watch?v=37sFIak42Sc, 2018

[44] Django, https://www.djangoproject.com

[45] Nzomo, Mbithe, Flask or Django: An In-Depth Comparison, https://scotch.io/bar-talk/flask-or-django-an-in-depth-comparison-part-one, 2018

[46] Kaggle, Sentiment140 Dataset, https://www.kaggle.com/kazanova/sentiment140

[47] Tathaga, Das, Zaharia, Matei and Wendell, Patrick, Diving into Apache Spark Streaming's Execution Model, DataBricks Engineering Blog, https://databricks.com/blog/2015/07/30/diving-into-apache-spark-streamings-execution-model.html, 2015

[48] Streaming Data Pipeline to Transform Store and Explore with Kafka Spark and Drill, https://dzone.com/articles/streaming-data-pipeline-to-transform-store-and-exp

[49] Apache Kafka Integration with Spark, https://www.tutorialspoint.com/apache_kafka/apache_kafka_integration_spark.htm

[50] MogoDB, NoSQL Vs. Relational Databases, https://www.mongodb.com/scale/nosql-vs-relational-databases

[51] Pulralsight, Developing Apache Spark Applications: Scala vs Python, https://www.pluralsight.com/blog/software-development/scala-vs-python

[52] Docker, What is a Container?, https://www.docker.com/resources/what-container