**Title to go here.**

**Abstract**

1. **Introduction**

The emergence of the internet can be traced back to the concept of a global computer network traced back to the 1960s. A paper published by Paul Baran, ‘On Distributed Communications’ is thought to be one of the foundational papers providing early inspiration for the design of the APRANET. The paper outlines the idea of a distributed communications network that could withstand partial outages. MIT researchers, including Lawrence Roberts, drew inspiration from Baran’s paper and proposed the initial design of the APRANET. In late 1969, the first APRANET message was sent between two nodes, which acted as a precursor to the modern internet. The internet, or the world-wide web (WWW) as it is sometimes coined, transformed from a communication between two nodes into a massive information exchange introduced by Tim Burners-Lee (T. Berners Lee, R. Cailliau, J. Groff, and B. Pollermann, “World wide web: the information universe,” Internet Res., vol. 20, no. 4, pp. 461–471, Aug. 2010.). The internet has now evolved into a significant medium for information sharing and communication. Communication systems via the internet can be traced back to developments such as Bulletin Board Systems (BBS) (1970s), UseNet (1979), and Internet Relay Chat (1988). These platforms primarily allowed for users to engage with each other via chat rooms, mail systems and the sharing of files. Significant developments emerged from 2000 onwards with regards to social media. Social media is defined as a ‘group of Internet-based applications that build on the ideological and technological foundations of Web 2.0, and that allow the creation and exchange of user-generated content’ (Kaplan and Haenlein, 2010). The introduction of Friendster in 2002 proved to be a significant turning point in social media platform structure by giving users control over whom they connect with (Winder, 2007). Facebook and Twitter revolutionised social media websites and became two of the most popular and everlasting platforms since the inception of the internet. This paper places a particular focus on Twitter by analysing the sentiment of a cohort of users during a particular interval. Twitter is defined by Mistry (2011) as being a free social networking tool that allows people to share information, in a real-time news feed through posting brief comments about their experiences and thoughts (Britsol et al 2010). Users broadcast short posts known as tweets which are limited to 280 characters including spaces and punctuation, appearing in followers’ feeds. Users can weave their post into specific topics by adding a hashtag to a keyword in their tweet. Twitter can also be used by businesses, entrepreneurs, journalists etc. to increase brand awareness and create a social media presence that can help; interact with customers, monitor public trends and make public announcements. As of December 2022, Twitter’s audience accounted for over 368 million monthly active users worldwide (Statista, 2024). Due to the high volume of attraction to the social media platform it can be extremely useful to derive analytical insights from the information posted. One such way of extracting useful insight from Twitter is to analyse the sentiment of user posts via Tweets on a particular subject matter. This paper provides a framework for which sentiment can be extracted from Twitter data using advanced data analytics methods and big data storage and processing techniques, including forecasting sentiment for future time periods outside of the original time series.

1. **Methodology**

2.1 Overview

The goal of the research paper was to identify and analyse historic sentiment from a large dataset gleaned from the Twitter API (‘ProjectTweets.csv’) using advanced data analytics techniques. Various sentiment types were explored including polarity, subjectivity and the Valence Aware Dictionary and Sentiment Reasoner (VADER). Time series analysis was applied to the sentiments in Python using various libraries and eventual forecasts were estimated for the period in question.

2.2 Data Storage and Processing

*2.2.1 Hadoop Database (HBase)*

The dataset contained 1,600,000 observations, each representing a tweet posted by a specific user at a particular time. Due to the size of the data and the task to be performed a distributed data processing environment was required to store and process the data. The data was stored in Hadoop Database (HBase). HBase is the Hadoop database. It is used when you need random, real-time, read/write access to your big data. HBase’s goal is the hosting of very large tables-billions of rows X millions of columns-atop clusters of commodity hardware (Apache Hbase). HBase is a NoSQL database built on top of Hadoop. NoSQL databases are databases by which data can be handled and extracted with ease. A NoSQL database is a type of database management system that allows for the storing and extracting of data in ways different to traditional RDBMS methods. HBase utilises Hadoop’s Distributed File System (HDFS) and MapReduce model and contrastingly stores data in columns as opposed to rows (Egan, 2017). HBase was chosen as a mean of storing data for this research paper because of its flexibility.

The ‘ProjectTweets.csv’ file was stored in the HDFS using the below command.

Once the file was correctly stored and uploaded in HDFS an empty table named **INSERT NAME HERE!!** was created in HBase using the below command.

This table was populated with the data from the ‘ProjectTweet.csv’ file using the below command. The below command had to be manipulated to create column names and maintain row numbers for each observation. Following the storage of ‘ProjectTweet.csv’ in HDFS and HBase the dataset was processed using Apache Spark. HDFS and Apache Spark are later utilised to process the text of the tweets for the creation of a Natural Language Processing machine in Python.

* + 1. *Apache Spark*

Apache Spark is a Java Virtual Machine (JVM) based on distributed data processing engine that scales, and it is fast compared to many other data processing frameworks. Spark, as it can be referred to is a framework for distributed computing which depends on Hadoop MapReduce algorithms. Unlike traditional MapReduce algorithms, Spark can store intermediary results using Memory Computing (Aaron N. Richter, Taghi M. Khoshgoftaar, Sara Landset, and Tawfiq Hasanin, “A Multi-Dimensional Comparison of Toolkits for Machine Learning with Big Data”, 2015 IEEE International Conference on Information Reuse and Integration (IRI), San Francisco CA, pp. 1-8, 2015). Spark provides APIs in Java, Scala, Python and R. For this research paper, the Python API is used, namely PySpark. PySpark is an API that allows for the creation of Spark applications using Python. PySpark was used to allow for parallel processing of data preparation and text processing problems to allow for accurate sentiment analysis.

Various user-defined functions (UDF) were created within PySpark to allow for sentiment analysis results with higher degree of accuracy. Notable deficiencies with the text in the ‘tweet’ column included usernames included in the tweet, url websites included in the tweet, special characters included in the tweet and unnecessary whitespace appearing. UDFs were applied on the tweet column using PySpark allowing for quick and efficient data preparation prior to sentiment analysis.

Following data preparation in PySpark, the final Spark table was exported to HDFS. Within HDFS the below command allowed for the Spark output file to be copy to the local drive on the Ubuntu Virtual Machine.

Due to the size of the file exported from HDFS to a local drive on the Ubuntu Virtual Machine, two separate csv files were created. These two files were then shared between the virtual machine and the host machine. Shared folders can be created between the Ubuntu Virtual Machine and the host machine to allow for the transfer of files between respective machines.

* 1. Text Processing and Exploratory Data Analysis

This section details the process of lemmatization, which is applied through Python functions on the tweet column of the dataset. Exploratory Data Analysis (EDA) is also performed to highlight initial observations of the key components of the data prior to sentiment analysis.

*2.3.1 Lemmatization*

Lemmatization is the process of assembling the inflected parts of a word such that they can be recognised as a single element, namely the word’s lemma or vocabulary form (Balakrishnan, Vimala & Ethel, Lloyd-Yemoh. (2014). Stemming and Lemmatization: A Comparison of Retrieval Performances. Lecture Notes on Software Engineering. 2. 262-267. 10.7763/LNSE.2014.V2.134..). Lemmatization follows a similar procedure to stemming however it maps a root word to a relevant word as opposed to a root stem. Lemmatization is utilised in this paper, as the lemma of a word is a real language text format and is good practice to apply to inputs prior to using Natural Language Processors (NLPs) such as sentiment analysis algorithms. The WordNetLemmatizer function is used from the NLTK module in Python to lemmatize text using WordNet’s built-in morphy function. This function returns the input word unchanged if it cannot be found in WordNet.

* + 1. *Exploratory Data Analysis*

The data used in this research project analyses tweet information which is categorical text data. There is no quantitative data in the raw dataset, prior to sentiment analysis. Included in this section are two charts, describing the number of tweets over time and a distribution of the number of charts.

A graph with numbers and lines

Description automatically generated

Preliminary observations can be drawn from the above chart. The chart is skewed left, meaning that there are a greater number of tweets included in the dataset in more recent time periods. This means that any sentiment analysis carried out in more recent months might be more statistically robust that earlier months due to a great number of observations. The period in the chart is short with date range spanning from April 2009 – June 2009. To increase the statistical robustness of any time series analysis and forecasting a longer period would be preferred. It can also be observed that there are certain dates where there are no tweets included in the dataset. An imputation will have to be carried out for dates with no tweet observations.

A graph of values and values

Description automatically generated

The above chart displays the character count distribution for the tweets included in the dataset. From the above we see that the character count is skewed right. This means that the most frequently occurring number of characters included in a tweet is lower than the average amount of characters included in a tweet. An interesting insight to include as part of this analysis would be to compare the distributions in character counts between the raw tweets included in the original dataset and the transformed tweets following some text processing.

* 1. Sentiment Analysis

The following section introduces the concept of sentiment analysis and the various methods used as part of this research paper in quantifying the sentiment from the tweets comprising the ‘ProjectTweets.csv’ dataset.

*2.4.1 Definition*

Sentiment analysis refers to the general method to extract polarity and subjectivity from semantic orientation which refers to the strength of words and polarity text of phrases (M.Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, “ LexiconBased Methods for Sentiment Analysis,” Association for Computational Linguistics, 2011). There are two main methods of performing sentiment analysis, the Lexicon-based method and the Machine learning based method. In this research paper three different sentiment methods are applied to the lemmatized tweet data. Polarity, subjectivity and VADER scores are assigned to tweets to quantify sentiment. The ‘TextBlob’ Python package is used to calculate polarity and subjectivity scores whereas the ‘SentimentIntensityAnalyser’ is used from the ‘VaderSentiment’ package in Python to calculate VADER scores for tweets.

* + 1. *Natural Language Processing (NLP) Methods*

Polarity and subjectivity are determined from text using NLP. Polarity is performed to ascertain the emotional attitude of text, classifying it as positive, negative or neutral (Vishal Kharde, Prof Sonawane, et al. 2016. Sentiment analysis of twitter data: A survey of techniques. arXiv preprint arXiv:1601.06971 (2016).). Subjectivity outlines whether a piece of text is opinionated or not opinionated (SAME REF). The ‘TextBlob’ Python package is utilized to perform polarity and subjectivity analysis on the tweets data. TextBlob is a popular tool that uses NLP to perform sentiment analysis on text data. Scores are assigned to observations for both polarity and subjectivity. The scores assigned for polarity range from -1 to +1, with -1 indicating a very strong negative opinion and +1 indicating a very strong positive opinion. The scores assigned for subjectivity range from 0 to 1, with 0 classified as very objective and 1 classified as very subjective.

A graph with red lines

Description automatically generated

The above chart indicates that most tweets are assigned a neutral sentiment. There appears to be an equal distribution between the positive and negative sentiment on either side of the neutral sentiment score.

A graph with red bars

Description automatically generated

The above chart indicates that most tweets are assigned an objective sentiment. There appears to be a concentration in tweets being assigned mid ranging scores 0.4 – 0.6 with another peak of tweets being assigned a score of 1 indicating subjectivity.

* + 1. *Lexicon Based Methods*

Lexicon-based methods make use of predefined lists of words where each word is associated with a specific sentiment (. Goncalves, F. Benevenuto, M. Araujo and M. Cha, “Comparing and Combining Sentiment Analysis Methods”, 2013.). VADER (Valence Aware Dictionary for Sentiment Reasoning) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. Hutto and Gilbert’s (2015) paper described the development, validation and evaluation of VADER, outlining advantages such as computational ease, accessible lexicons and the generality used allows VADER to be self-contained and domain agnostic (i.e does not require large amounts of training data). VADER informs users about the positivity or negativity score along with the extent to which a piece of text is either positive or negative. The compound score produced from the VaderSentiment Python package is reported in this paper, which is a score that ranges from -1 to 1, with -1 as negative and 1 as positive.

A graph with red bars

Description automatically generated

The above chart is like the polarity distribution of scores. The scores are centralised at neutral scores and display a similar distribution on either side of neutral. There appears to be an even distribution between positive and negative sentiment according to the VADER scores.

1. **Time Series Analysis**

3.1 Overview

This section describes the process for incorporating time series analysis and forecasts applied to the sentiment analysis results previously detailed. The models explored and evaluated as part of this research paper for forecasting Twitter sentiment include a Random Forest regressor, Autoregressive Integrated Moving Average (ARIMA) and a Recurrent Neural Network. To create a time series, the sentiment data had to be aggregated. A daily time series was created, aggregating the average sentiment values for each sentiment category. This resulted in 47 observations from April 2009 to June 2009. This is a very low aggregation of data to build robust time series models and an alternative method to overcome this would be to aggregated hourly data and forecast future values hourly and thus this acts as a drawback to the results of this research paper. There were also missing dates in between observations which were forward filled. The final dataset contained 81 observations on which to build models and forecast future sentiment.

* 1. Forecast Models Methodology

The following section describes the build of the models used for forecasting purposes and final forecast values for 1-week, 1-month and 3-month intervals for each sentiment type. Models are fit by splitting the time series into training and testing datasets. This allows for out of sample testing to tune hyperparameters to be used in the final forecasts.

*3.2.1 Random Forest Regression Model*

A Random Forest is a tree-based ensemble method with each tree depending on a collection of random variables. Random Forests can be used for either a categorical or continuous response variable. (**INSERT**). Random forests are relatively fast to train and predict and naturally handle regression problems. The ‘RandomForestRegressor’ function from scikit-learn in Python is used to forecast future sentiment. According to Pedregosa et al (2011), a random forest is a meta estimator that fits several classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The ‘ForecasterAutoreg’ is wrapped around the random forest model creating pre-defined lags of the time series value to be included as the predictor variables. The arguments for the random forest and the lags to use for the forecast model are tuned using the ‘grid\_search\_forecaster’ function from the Skforecast library. The number of lags to include, the maximum depth of the trees in the random forest and the number of trees in the forest are optimally tuned for each sentiment type.

3.2.1.1 Random Forest Regression Parameter Results

|  |  |  |  |
| --- | --- | --- | --- |
| **Values** | **Parameters** | | |
| **Sentiment** | **Max Depth** | **Number of Estimators** | **Lags** |
| Polarity | 3 | 100 | 20 |
| Subjectivity | 3 | 100 | 10 |
| VADER | 3 | 500 | 10 |

The maximum depth of the trees used for each sentiment is 3. The number of trees in the random forest are 100 for both polarity and subjectivity, with the number of trees for VADER sentiment being 500. 20 days are lagged for polarity and 10 days are lagged for subjectivity and VADER. These are the final model parameters used for the forecasts, reported in the final results section.

* + 1. *ARIMA Model*

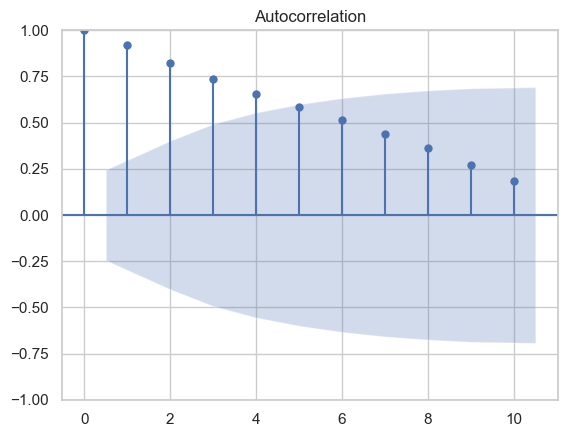
Autoregressive integrated moving average (ARIMA) modelling is a type of univariate modelling, where a time series is expressed by its own past values (the autoregressive components) along with current and lagged values of a ‘white noise’ error term (the moving average component) (**CBI**). ARIMA modelling was selected as part of this research as the time series itself is the only required component, unlike multivariate regression where exogenous variables are required. The notation used for ARIMA models is (p,d,q), where p denotes the number of autoregressive terms, d denotes the number of times the series is difference for stationarity and q denotes the number of moving average terms.

The first step for ARIMA fit is to test the time series for stationarity. The Augmented Dicky-Fuller (ADF) test is a statistical procedure that tests stationarity. The ‘adfuller’ function in the statsmodel Python library is used to perform the ADF test on polarity, subjectivity and VADER sentiment. The null hypothesis of the ADF test is that there is a unit root in the time series. The p-value of this test is reported and if the value is above a critical size (5%) then a unit root is present, and the time series must be differenced.

The second step involved selecting the p, d, and q elements of the model by plotting the Autocorrelation and Partial Autocorrelation plots. The below ruleset should be considered when selecting the model structure for ARIMA.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **AR(p)** | **MA(q)** | **ARMA(p,q)** |
| **ACF** | Tails off | Cuts off after lag q | Tails off |
| **PACF** | Cuts off after lag p | Tails off | Tails off |

An example of the ACF and PACF can be seen below for the polarity sentiment explored in the time series.

A graph with a line graph

Description automatically generated

Using the aforementioned ruleset, the order of the model is an AR(2). This was fit and used for the various forecasts for the polarity sentiment.

* + - 1. *ARIMA Model Orders*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **ARIMA (p,d,q)** | | |
| **Sentiment** | **p** | **d** | **q** |
| Polarity | 2 | 0 | 0 |
| Subjectivity | 3 | 1 | 1 |
| VADER | 2 | 0 | 0 |

The above table reports the orders used for the p, d, and q elements for the different sentiment scores used in the time series. These orders correspond to the final models used for sentiment forecasts.

1. **Results & Conclusion**