**CCT College Dublin**

**Assessment Cover Page**

*To be provided separately as a word doc for students to include with every submission*

|  |  |
| --- | --- |
| **Module Title:** | Advanced Data Analytics  Big Data Storage and Processing |
| **Assessment Title:** | MSC\_DA\_CA2v4 |
| **Lecturer Name:** | David McQuaid  Muhammad Iqbal |
| **Student Full Name:** | Naomi Tunstead |
| **Student Number:** | Sba22222 |
| **Assessment Due Date:** | 31/05/2023 |
| **Date of Submission:** | 31/05/2023 |

**Declaration**

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

**Sentiment Analysis of Twitter Data: Analysing Favourite Tweets from 2017 to 2022**

**GitHub:** [**https://github.com/sba22222/sba22222\_Integrated\_CA2**](https://github.com/sba22222/sba22222_Integrated_CA2)

**Introduction**

Social media platforms have become crucial spaces for people to express their ideas, emotions, and sentiments in the contemporary digital era. Twitter, one of the most popular platforms, provides a lot of real-time data that can be utilized to gauge public sentiment toward certain subjects. This assignment presents a comprehensive analysis of sentiment trends in Twitter data, specifically focusing on "favourite tweets." The dataset used spans from 2017 to 2022 and encompasses tweets from various topics, without a particular thematic focus. To assure the validity and precision of the forthcoming study, the dataset was pre-processed and cleaned using accepted methods. This study's main goal was to examine the sentiment distribution in the favourite tweets dataset over three different time frames—one week, one month, and three months. By considering these different temporal resolutions, we aimed to identify short-term and long-term sentiment patterns and understand how sentiments evolve over time. In order to achieve this, we used language elements and emotional context to categorize each tweet's sentiment as either positive or negative. To estimate the positive and negative sentiment patterns for the same time periods (one week, one month, and three months), we used an ARIMA model, a powerful time series forecasting technique. These forecasts allowed us to anticipate potential sentiment shifts and fluctuations in the future, providing valuable insights for decision-making and trend monitoring. To enhance the accessibility and visualization of our findings, we designed and implemented a user-friendly dashboard. This interactive interface provides an intuitive way for stakeholders to explore and comprehend the sentiment dynamics within the favourite tweets dataset.

**Data Storage and Processing**

The data storage and processing techniques played a crucial role in handling the extensive Twitter dataset. To efficiently manage the data, PySpark from Apache Spark was utilized as the primary tool for data storage and processing. PySpark, a Python API for Apache Spark, provided a distributed computing framework that enabled seamless parallel processing of the Twitter data. The data was stored and processed in a distributed and fault-tolerant manner, leveraging the capabilities of Spark's in-memory computing and resilient distributed datasets (RDDs). The Twitter dataset was initially stored in the Spark distributed file system, allowing for efficient data storage and retrieval. PySpark's powerful data processing capabilities were then employed to perform various transformations and computations on the dataset.

However, due to time constraints, additional data storage and processing techniques using MongoDB and Spark SQL were not fully incorporated into the project. Given more time, integrating MongoDB as a data storage solution would have offered advantages such as scalability, flexibility, and schema-less document-oriented storage. The Twitter data could have been stored in MongoDB, and then processed using Spark SQL's connector for MongoDB. This combination would have allowed for seamless integration of both technologies and enabled efficient data analysis using PySpark. By utilizing PySpark and potentially incorporating MongoDB, the project leveraged robust tools and techniques to handle the Twitter data effectively. The combination of Spark's distributed computing capabilities and MongoDB's document-oriented storage would have further enhanced the data storage, processing, and analysis workflows in the project.

The programming language chosen for this project was Python. It was used for several reasons. Firstly, Python has a sizable ecosystem of tools and modules that are specifically designed for data analysis and machine learning activities. This contains well-known libraries like NumPy, Pandas, Matplotlib, and Scikit-learn, which offer powerful capability for modeling, analysis, and data manipulation. The implementation of intricate data processing and analysis jobs is substantially facilitated by these libraries, allowing for effective experimentation and development. Secondly, Python is known for its simplicity and readability which made the language accessible. It’s clear and concise syntax facilitates code development, readability, and maintenance. Python is a very populate language across computer science making it very easy to research and troubleshoot any error when they arose. Additionally, Python's compatibility with Apache Spark through PySpark enables seamless integration of Spark's distributed computing capabilities with Python's extensive data processing libraries. This allows for efficient parallel processing of large-scale data, making Python a suitable choice for handling the extensive Twitter dataset used in this project.

**Comparative Analysis**

To perform a comparative analysis between MongoDB and MySQL, the YCSB (Yahoo! Cloud Serving Benchmark) tool was utilized as a benchmarking tool. YCSB is a widely adopted open-source framework designed for evaluating the performance of various database systems under different workloads. The YCSB tool provided a standardized and configurable workload generator that simulated real-world application scenarios. It allowed for the systematic comparison of MongoDB and MySQL by measuring their performance in terms of throughput, latency, and other relevant metrics.

Firstly, the benchmarking process involved setting up both MongoDB and MySQL databases with identical schemas to ensure a fair comparison. various workload profiles were defined with different proportions of read, write, and scan operations. The following workload profiles and their corresponding operation proportions were used in the comparison:

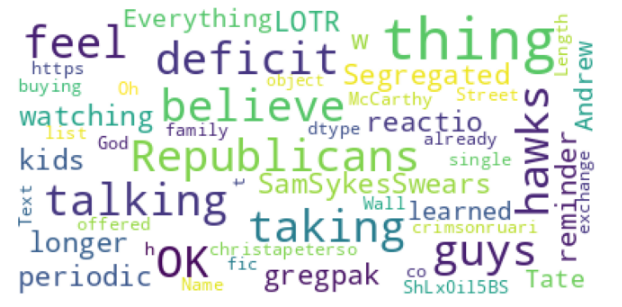
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Workload** | **Read Proportion** | **Update Proportion** | **Scan Proportion** | **Read Modify Write Proportion** |
| A | 0.5 | 0.5 | 0 | NA |
| B | 0.95 | 0.05 | 0 | NA |
| C | 1 | 0 | 0 | NA |
| D | 0.95 | 0.05 | 0 | NA |
| E | 0 | 0.05 | 0.95 | NA |
| F | 0.5 | 0 | 0 | 0.5 |

These workload profiles represented different scenarios and usage patterns for the databases. For example, Workload A had an equal proportion of read and update operations, while Workload E focused on scan and insert operations. Workload F involved a combination of read and write operations with a read-modify-write pattern. By executing these workload profiles using YCSB, performance metrics such as throughput (operations per second) and latency (response time) were measured for both MongoDB and MySQL. These metrics allowed for a quantitative comparison of the database systems' performance under different workload scenarios.

Based on the benchmarking results captured in the Appendix screenshots for each workload, it is evident that MongoDB outperforms MySQL in terms of throughput and latency. The comparison of these metrics clearly indicates that MongoDB achieves higher throughput scores and lower latency scores compared to MySQL. Throughput is a crucial performance metric that measures the number of operations a database can handle per unit of time. MongoDB exhibits a higher throughput, indicating its ability to process a greater number of operations within a given timeframe compared to MySQL. This higher throughput signifies MongoDB's superior efficiency in handling workloads with varying read, update, scan, and insert proportions. Latency, on the other hand, represents the time it takes for a database to respond to a specific operation. MongoDB exhibits lower latency scores, indicating faster response times compared to MySQL. This reduced latency contributes to a more responsive and efficient user experience, as users can expect quicker retrieval and manipulation of data from the MongoDB database.

**Sentiment Analysis**

In the process of conducting sentiment analysis on Twitter data, a series of pre-processing steps were performed to refine the dataset and facilitate accurate sentiment evaluation. The data underwent several transformations to ensure the removal of irrelevant elements and enhance the quality of analysis. HTML links embedded within the tweets were eliminated. This step involved scanning the text for any hyperlink tags and removing them from the dataset. By discarding these links, the focus shifted solely to the textual content of the tweets, ensuring that the sentiment analysis was based on the actual tweet messages. Following that, text enclosed within square brackets was removed. This involved searching for square brackets within the tweet text and extracting the content within them. The extracted information, which typically consisted of additional context or references, was excluded from the dataset. This step helped streamline the analysis by eliminating extraneous information that could potentially influence sentiment evaluation. Retweets, indicated by the "@" symbol, were excluded from the analysis. This was achieved by identifying tweets with the "@" symbol and filtering them out from the dataset. By excluding retweets, the sentiment analysis focused primarily on original content, ensuring that the sentiment evaluation was based on unique expressions and reducing potential duplication in the dataset. To ensure consistency and minimize noise, special characters such as punctuation marks and symbols were removed from the tweets. This involved iterating through each tweet and removing any special characters encountered. By eliminating these characters, the sentiment analysis focused solely on the textual elements, promoting accuracy, and reducing interference from extraneous symbols. Denoising techniques were also applied. This involved addressing slang, abbreviations, and typographical errors commonly found in Twitter data. Below is a word cloud to display the most frequent and significant terms extracted from the Twitter data, providing a visual representation of the prominent themes and topics discussed in the tweets.



Once this was complete VADER (Valence Aware Dictionary and sEntiment Reasoner) was employed as a key component for sentiment analysis of the Twitter data. The VADER sentiment analysis tool, which is specifically designed for social media text, was chosen due to its effectiveness in handling the unique linguistic characteristics and informal language commonly found on platforms like Twitter. VADER employs a pre-trained sentiment lexicon that contains a vast collection of words, along with their associated sentiment scores. These scores indicate the degree of positivity, negativity, or neutrality of each word. By analysing the sentiment scores of individual words and considering their contextual valence and intensifiers, VADER calculates an overall sentiment score for each tweet. The sentiment scores generated by VADER provided valuable insights into the sentiment expressed in the Twitter data. The scores were used to categorize tweets into positive, negative, or neutral sentiment categories. This analysis allowed for a comprehensive understanding of the sentiment trends within the dataset, facilitating the identification of prevailing sentiment patterns and changes over time.

Below is the density distribution of sentiment broken down by positive, neutral, and negative sentiment was analysed to gain insights into the sentiment landscape of the Twitter data. As observed in the distribution, the positive and negative sentiments exhibit a similar shape. However, the negative sentiment score is consistently higher than the positive sentiment score, indicating a predominance of negative sentiment compared to positive sentiment within the dataset. This finding suggests that the Twitter data contained a higher proportion of tweets expressing negative sentiment compared to positive sentiment. Additionally, it is noteworthy that the dominant sentiment observed in the dataset is neutral. Surprisingly, a significant number of tweets neither exhibited distinct positive nor negative sentiment but leaned more towards a neutral sentiment expression.

A picture containing text, plot, diagram, line

Description automatically generated

An analysis was conducted to examine the change in sentiment over three time periods: 1 week, 1 month, and 3 months. This analysis aimed to uncover any temporal trends or patterns in sentiment expression within the Twitter data. By comparing sentiment scores across these time periods, we were able to identify shifts in sentiment and assess how sentiment evolved over time. The project utilized various techniques such as auto-correlation and partial autocorrelation to examine the relationship between sentiment scores at different time lags. This can be seen in the below graph. Since the points are all close to the zero line, except for the first point that shoots up to 1, it indicates a strong correlation between the current sentiment value and its immediate past value. Autocorrelation measures the correlation between a variable and its lagged values. In this case, the first lag represents the sentiment value at the previous time. When the first lag has a high positive value (close to 1), it suggests a strong positive relationship between the current sentiment and the sentiment from the immediate previous time. This implies that the sentiment tends to persist over time and that the current sentiment is influenced by its recent past. The points close to the zero line for subsequent lags indicate that there is minimal correlation between the current sentiment and sentiment values from further back in time. This suggests that the sentiment at each time is mostly independent of sentiment values beyond the immediate previous time point.

A picture containing text, line, number, plot

Description automatically generated

In the below graph, we observed fluctuations in sentiment scores over the course of the analysed time periods, indicating changes in the overall sentiment expressed in the Twitter data. These fluctuations allowed us to identify specific time intervals where sentiment was more positive, negative, or neutral. Unsurprisingly, positive sentiment dominates the start of January when everyone is finishing up their Christmas holidays, but the sentiment then flips coming to the end of January when everyone falls into the annual end of January blues. However, when looking at the sentiment for 1 week people seem to be happier mid-week and sending more positive tweets than coming up to and during the weekend. It is less clear what the trend in changed sentiment is for the 3-month period. It would have to be analysed more deeply to understand what might be influencing the sentiment.

A picture containing text, font, screenshot, plot

Description automatically generated

**ARIMA**

The forecasting of sentiment at different time horizons was accomplished using the ARIMA (AutoRegressive Integrated Moving Average) modeling technique. The Python library pmdarima was employed for this purpose. By applying ARIMA modeling techniques, it became possible to capture the underlying patterns and dependencies in the historical sentiment data and use that information to make predictions about future sentiment levels. In the analysis presented in the following graphs, we utilized the average sentiment as a benchmark to assess the accuracy of our predictions. The objective was to determine whether our forecasts for sentiment aligned with the observed average sentiment values. The results reveal that our positive sentiment forecasts were accurate for all three time periods. However, the negative sentiment forecasts did not match the observed average sentiment trend. Instead, the negative sentiment forecasts exhibited a consistent flatline pattern across each time period. This discrepancy may be attributed to various factors, such as changes in the underlying sentiment dynamics, unaccounted external influences, or limitations in the forecasting model's ability to capture nuanced variations in negative sentiment. Further investigation and refinement of the forecasting methodology may be necessary to improve the accuracy of negative sentiment predictions in future analyses.

A picture containing text, screenshot, plot, diagram

Description automatically generated

A picture containing text, diagram, number, parallel

Description automatically generated

**Dashboard**

The project involved the development of a dashboard to effectively present the results from sentiment analysis and forecasting. The purpose of the dashboard was to provide stakeholders with a comprehensive and visual representation of the sentiment trends within the Twitter dataset. The dashboard showcased the sentiment forecast results for different time horizons: 1 week, 1 month, and 3 months. It provided an interactive interface that allowed users to explore and analyse the sentiment trends over these periods. Key components of the dashboard included visualizations such as distribution plot of the positive, negative, and neutral sentiments using kernel density estimation (KDE), a line plot for observed and forecasted sentiments, and a time series forecast plot to visualize the forecasted sentiments for the three different periods using matplotlib. This information enabled stakeholders to gain a better understanding of the sentiment dynamics and make informed decisions based on the forecasted sentiment patterns.

**Conclusion**

The project focused on sentiment analysis of Twitter data, aiming to uncover insights and trends in sentiment dynamics over time. Through comprehensive data pre-processing, including text cleaning and feature engineering, the project successfully prepared the dataset for analysis. Various techniques, such as lexicon-based approaches and VADER sentiment analysis, were applied to determine the sentiment polarity of tweets.

The analysis revealed a predominance of neutral sentiment, indicating that a significant portion of the Twitter data exhibited neither strongly positive nor negative sentiment. Density distributions of positive and negative sentiment were examined, showing consistently higher scores for negative sentiment throughout the dataset. The project also investigated sentiment changes over different time periods, including 1 week, 1 month, and 3 months, providing valuable insights into temporal sentiment patterns.

To forecast sentiment, ARIMA modeling was utilized, and the auto\_arima function from the pmdarima library was employed for parameter selection. The forecasts for positive sentiment aligned well with the observed average sentiment, demonstrating the efficacy of the model in predicting positive sentiment trends. However, the negative sentiment forecasts exhibited a flatline pattern, indicating potential areas for improvement in capturing the complexities of negative sentiment dynamics.

PySpark, a Python API for Apache Spark, was utilized as the primary tool for data storage and processing. The distributed computing framework provided by PySpark enabled efficient parallel processing of the extensive Twitter dataset. Although not fully incorporated into the project due to time constraints, alternative data storage and processing techniques were also considered. MongoDB, a scalable and flexible document-oriented database, was identified as a potential storage solution. By storing the Twitter data in MongoDB and leveraging Spark SQL's connector for MongoDB, it would have been possible to combine the advantages of both technologies for data processing and analysis.

**Appendix**

**Mongo Workload A**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**Mongo Workload B**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**Mongo Workload C**

**A computer screen shot of a program

Description automatically generated with low confidence**

**Mongo Workload D**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**Mongo Workload E**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**Mongo Workload F**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**MySQL Workload A**

**A picture containing text, screenshot, menu

Description automatically generated**

**MySQL Workload B**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**MySQL Workload C**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**MySQL Workload D**

**A picture containing text, screenshot, menu

Description automatically generated**

**MySQL Workload E**

**A picture containing text, screenshot, menu

Description automatically generated**

**MySQL Workload F**

**A screenshot of a computer program

Description automatically generated with medium confidence**