# Home - CCT College Dublin

# Group ID - MSc in Data Analytics

**Autor: Joao Ribeiro**

e-mail:[sba22203@student.cct.ie](mailto:sba22203@student.cct.ie)

Student ID: sba22203

*Advanced Data Analytics / Big Data Storage and Processing*

Title: Sentiment analyses on favorite-tweets

**Dublin, 24 May 2023**

## **Acknowledgments**

To my teachers David McQuaid and Muhammad Iqbal from the Master of science in Data Analytics at CCT college.

Word Count: 3297

Github: https://github.com/sba22203

**Introduction**

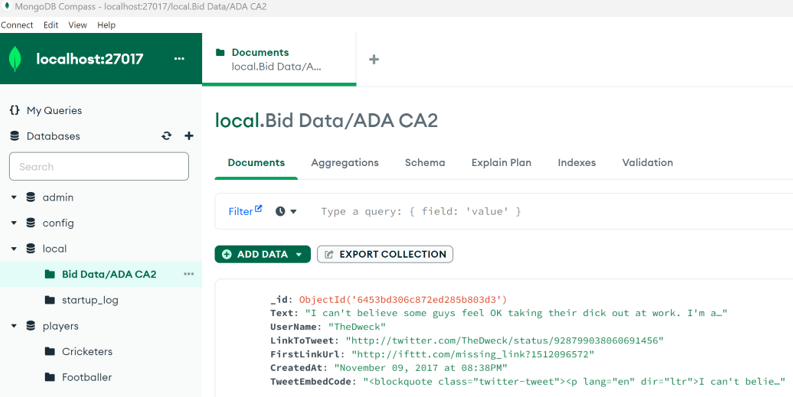
For my study, I tried several times to scrap tweets, but not working keys access, so I chose an existing dataset (favorite-tweets.json) from [a2liz/favorited-tweets | Workspace | data.world](https://data.world/a2liz/favorited-tweets/workspace/file?filename=favorite-tweets.jsonl), with a licence of public domain: the advantages of the dataset is that it covers several years (from November 2017 to January 2023), making an excellent choice for sentiment and to forecast time series. The tweets are also good to pre-process and realize the EDA.

1. **Data storage**
   1. MongoDB

I’ve chosen MongoDB firstly, its flexible data model allows for handling unstructured data efficiently (Chodorow & Dirolf, 2013); secondly, MongoDB's high-performance capabilities enable real-time data processing and analysis (Banks, 2014); thirdly, its scalability allows for the seamless handling of increasing data volumes without compromising performance (Copeland, 2010) and lastly, MongoDB's rich query language supports complex queries and aggregations, including full-text search and geospatial queries (Chodorow & Dirolf, 2013).

These benefits make MongoDB an excellent choice for sentiment and time series analysis with the "favorite-tweets.jsonl" dataset, offering easy data ingestion, ad-hoc querying, and efficient storage of unstructured social media data (Banks, 2014; Rashid, 2013).

Let’s upload my JSON file to MongoDB Compass (fig. 1).

****

*fig. 1*

Let’s export into an excel file “ADA CA2” (fig. 2*)*



*fig. 2*

1. Couchbase

Couchbase offers several key benefits as a NoSQL database solution. Firstly, it provides high performance and scalability, allowing for efficient handling of large volumes of data and high-speed data retrieval. Secondly, it offers flexible data modeling, supporting a variety of data structures and enabling agile development. Thirdly, it ensures high availability and fault tolerance with built-in replication and automatic failover mechanisms. Additionally, Couchbase's advanced caching mechanisms optimize data access, reducing latency and improving overall application performance. Lastly, its powerful querying capabilities and seamless integration with popular programming languages make it developer-friendly and adaptable to diverse use cases.

Let’s import the JSON file to Couchbase (fig. 3).

A screenshot of a computer

Description automatically generated with medium confidence

*fig. 3*

1. **Data processing in a Spark environment**

My main choices of Databricks over VM are:

Ease of use: Databricks offers a managed Spark environment with automated cluster setup and maintenance. It provides a user-friendly web interface and collaboration features. In contrast, setting up Spark on a virtual machine requires manual installation, configuration, and management, which can be more complex and time-consuming.

Scalability: Databricks offers auto-scaling, which adjusts cluster size based on workload. It enables seamless scaling up or down as needed, optimizing resource utilization. In a virtual machine setup, the cluster is manually scaled by provisioning additional VMs or adjusting specifications, which is more cumbersome and less efficient.

* Spark SQL

Let’s import my CSV file to Databricks to use Apache Spark SQL, some data processing, MapReduce feature and data visualizations (fig. 4).

**A screenshot of a computer

Description automatically generated with low confidence***fig. 4*

The data type (fig. 5) is string.

**Uma imagem com texto, captura de ecrã, número, Tipo de letra

Descrição gerada automaticamente**

fig. 5

**Pre-process data**

Now I’ll do a simple process data, by converting the values in the *CreatedAt* column from a string representation of a date and time to a proper datetime format. This pre-processing step is necessary to make the data suitable for further analysis or processing, as the *CreatedAt* column is currently stored as a string and cannot be easily used for time-based analysis or queries (fig. 6).

A screenshot of a computer

Description automatically generated with low confidence

(fig. 6)

*MapReduce*

The MapReduce process is a programming model and framework created to handle extensive data processing and analysis in a distributed computing environment. It enables effective data processing by dividing it into smaller, manageable parts that can be processed concurrently across a cluster of computers.

the MapReduce process I used involves the following steps:

* Input Splitting: The initial dataset, containing tweets and corresponding usernames, would be divided into smaller subsets called input splits. These splits enable parallel processing by assigning each split to a separate node in the cluster.
* Mapping: Each node processes its assigned input split independently. In this case, the mapping phase would involve extracting the username from each tweet and emitting a key-value pair where the username is the key and a constant value (e.g., 1) is the value.
* Shuffling: The map output is then shuffled and sorted based on the username key. This step ensures that all values associated with a particular username are grouped. It facilitates the subsequent reduction phase.
* Reducing: Each node receives the shuffled data and performs the reduction phase independently. The reducer combines the values associated with each username, in this case, by counting the occurrences of the constant value.
* Final Output: The reducer outputs the result, which includes the unique usernames and their corresponding tweet counts. The results are typically sorted in the desired order, in this case, in descending order of the tweet count and limited to 12 top tweets (fig. 7).

A screenshot of a computer

Description automatically generated with medium confidence(fig. 7)

now let’s visualize the results.

The bar chart (fig. 8) allows us to compare username’s *tweetscounts* and we can see *OhNoSheTwitnt* (first), *saladinahmed* (second place).

**A picture containing text, diagram, screenshot, line

Description automatically generated**

(fig. 8)

The pie chart (fig. 9) is interesting and permits us to check the weight of every user’s tweet count; here we can see the 2 highest:

*OhNoSheTwitnt* with 183 tweets (represents 17% of total tweets, and *Saladinahmed* with 165 tweets which represents 15.3%. Both represent 32.3% of the total (17+15.3).

**A picture containing text, electronics, compact disk, circle

Description automatically generated**

(fig. 9)

* Pyspark

Exploratory Data Analysis

The *describe* function allows us to get some information about the columns (6), the row number (4969 for all), mean and standard deviation are null since we’re dealing with strings, minimum and maximum information for each column (fig. 10).

A picture containing text, line, font, number

Description automatically generated  
(fig. 10)

There are 46 duplicates in my dataset (fig. 11) and will remove them.

A picture containing text, font, white, design

Description automatically generated

(fig. 11)

Now let’s check unique values (fig. 12).

A picture containing text, font, screenshot, line

Description automatically generated

(fig. 12)

The column *CreatedAt* has the lowest unique value of 1386 (lots of duplicates), and the highest is *LinkToTweet* with 4850, out of 4969 rows in total.

Now let’s find Null values (fig. 13)

A picture containing text, screenshot, font, line

Description automatically generated

(fig. 13)

I have a great dataset with no null values, so I don’t need to process them.

Let’s export the processed dataset in Excel “ADA-BigData CA2” to import to Jupiter notebook and prepare sentiment and time-series analysis.

1. **Sentiment analysis**
2. Pre-processing

I’ll do some pre-processing steps that aim to clean and normalize the text data in the *Text* column, preparing it for further analysis in Sentiment and time series.

* Handle missing values: any Naan (missing) values in the 'Text' column are filled with empty strings.
* Remove Twitter handles: Strings starting with '@' followed by non-whitespace characters are removed from the 'Text' column.
* Remove hashtags: Strings starting with '#' followed by non-whitespace characters are removed from the 'Text' column.
* Remove URLs: Strings starting with 'http' followed by non-whitespace characters are removed from the 'Text' column.
* Remove special characters: All special characters are removed from the 'Text' column, and the remaining words are joined with spaces.
* Remove single characters: Single characters surrounded by whitespace are removed from the 'Text' column.
* Substitute multiple spaces: Multiple spaces in the 'Text' column are substituted with a single space (ignoring case sensitivity).
* Vader Sentiment Analysis

VADER is a reliable sentiment analysis tool that uses a dictionary-based approach to determine sentiment in text. It offers several advantages over other methods, including accuracy in analysing social media texts, efficient processing, detailed multi-dimensional analysis, and the ability to detect specific emotions. Unlike some approaches, VADER doesn't require training data for reliable sentiment analysis.

* EDA

Now let's visualize the distribution (fig. 14) and cumulative density (fig. 15) of sentiments across the tweets in the dataset.

A picture containing text, plot, screenshot, line

Description automatically generated

(fig. 14)

The depicted plot showcases the sentiment value distribution for negative, positive, and neutral tweets along the x-axis, ranging from -1 (most negative sentiment) to 1 (most positive sentiment). The y-axis represents the density of tweets associated with each sentiment value.

This plot allows us to understand how the sentiment values are distributed across the dataset. In this case, we can observe that the sentiment distribution appears to be relatively balanced, with a similar density of tweets for negative, positive, and neutral sentiments. It follows a normal distribution with similar distribution between negative and positive tweets (meaning no significant differences in the strength). Most positive tweets (1.0) have neutral sentiment.

A picture containing text, line, plot, screenshot

Description automatically generated

(fig. 15)

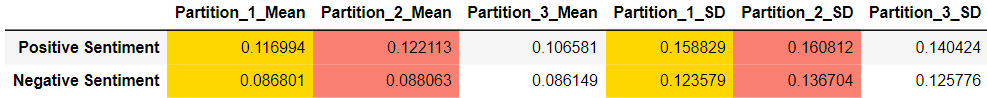
The plot showcases the cumulative distribution of sentiment values for negative, positive, and neutral tweets. The x-axis represents sentiment values ranging from -1 to 1. The y-axis represents the cumulative density of tweets falling within each sentiment value. The plot utilizes a KDE to estimate the cumulative distribution function (CDF) of the sentiment values. This plot helps determine the percentage of tweets that fall into each sentiment category. From the plot, we can observe that around 60% of the tweets are classified as neutral, followed by approximately 20% for both positive and negative tweets. The CDF plot offers a more detailed view of the sentiment distribution and assists in identifying any outliers or extreme values.

* Time-Based Analysis

It appears that the sentiments do not exhibit stationarity, meaning they do not have a constant mean and variance. Let’s test this hypothesis by dividing the data into three partitions. The question arises whether this lack of stationarity suggests the presence of a trend in the data.

* statistical measures calculated for each sentiment and partition.

Let’s calculate the means and standard deviations of positive and negative sentiment values for three different partitions of a dataset (fig. 16). The dataset is divided into three partitions based on index ranges.



(fig. 16)

These values represent the statistical measures calculated for each sentiment and partition. The mean values indicate the average sentiment value for each partition, where higher values denote a more positive sentiment. The standard deviation values represent the spread or variability of sentiment values within each partition, where larger values suggest more dispersed sentiment data.

Positive Sentiment: Partition 2 has the highest mean value (the most positive sentiment) followed by Partition 1 and then 3.

Negative Sentiment: All three partitions have relatively similar mean values, indicating a similar level of negative sentiment.

Positive Sentiment Standard Deviation: Partition 2 has the highest standard deviation, indicating greater variability in positive sentiment values compared to the other partitions.

Negative Sentiment Standard Deviation: Partition 2 also has the highest standard deviation, suggesting more variability in negative sentiment values within that partition.

These comparisons give insights into how positive and negative sentiments vary across the partitions. Partition 2 generally stands out with higher mean and standard deviation values for both positive and negative sentiment, suggesting a more diverse range of sentiment in that partition compared to the others.

* Plots (fig. 17)

A picture containing text, screenshot, font, plot

Description automatically generated

(fig. 17)

We could think it’s a stationary time series, with cyclic behaviour on a horizontal trend, whose statistical properties are independent over time, but it has a trend, making it a non-stationary time series.

* Time series forecast period one week, one month and three months.

We calculate the values for each period (fig. 18)

A screenshot of a computer

Description automatically generated with medium confidence

(fig. 18)

* forecasted values plots (fig. 19).

A picture containing text, screenshot, plot, line

Description automatically generated

(fig. 19)

We can see that for the forecast of three periods in 2023, positive and negative sentiments have high and low fluctuations, but follow a horizontal line.

The highest positive sentiment on one week is 03 Jan (0.4277), one month is 08 Jan. (0.498) and three months 18 Jan. (0.498). The lowest negative sentiment on one week is 04 Jan. (0.075), one month is 30 Jan (0.055), and three months 03 Mar (0.05).

* Time series decomposition (fig. 20)

Decomposition is commonly employed to eliminate the seasonal effect from a time series dataset, resulting in a clearer representation of underlying trends and patterns. This technique helps in better understanding the inherent characteristics of the data.

A picture containing text, screenshot, font, plot

Description automatically generated

(fig. 20)

Those plots aim to break down the sentiment data into its trend, level, seasonal, and residual components using seasonal decomposition, separately for positive and negative sentiments, allowing for a better understanding of its underlying patterns. The resulting graph can facilitate the detection of any seasonal patterns in the sentiment data and provide insight into the overall trend of the data.

By the plots, we can see no seasonality. The observed values, the trend (a value that causes variation pattern in time series), and the residual (noise) is showing a high variability.

We can confirm that it’s a non-stationary time series because it has a trend that will affect the mean, variance, and other properties at a certain point in time.

* Autocorrelation

The plot aims to conduct autocorrelation analysis on the positive and negative sentiment data. Autocorrelation analysis helps identify patterns and dependencies within the data by measuring the correlation between the data and its lagged values (fig. 21).

**A close-up of a graph

Description automatically generated with low confidence**

(fig. 21)

By examining the resulting plot, we can identify significant autocorrelation patterns in the sentiment data. Positive peaks or valleys in the autocorrelation plot indicate notable correlations at specific lags, suggesting the presence of recurring patterns or trends in the data.

Now let’s create plots to visualize the autocorrelation function (fig. 22) and partial autocorrelation function (fig. 23) of the positive and negative sentiment data.

The ACF helps us understand the correlation between the sentiment data and its previous values at different time lags. On the other hand, the PACF considers only the direct correlation between the sentiment data and its lagged values, excluding any indirect correlations through intervening values. These plots provide insights into the relationships and patterns in the sentiment data over time.

**A picture containing text, plot, line, screenshot

Description automatically generated**

(fig. 22)

**A picture containing text, plot, line, screenshot

Description automatically generated**

(fig. 23)

When examining ACF and PACF plots, researchers can identify relevant autocorrelation patterns in the data (Hyndman & Athanasopoulos, 2018).

The ACF plots reveal significant correlations at specific lags, while the PACF plots assist in determining the number of significant lags present in the data (Hyndman & Athanasopoulos, 2018).

To ensure clear visibility of the correlation values, the y-axis limit for each subplot in the plot is set from -1.1 to 1.1 (Hyndman & Athanasopoulos, 2018). This approach facilitates the identification of the appropriate lag order for time series models applied to sentiment data.

In our case, the ACF and PACF plots show autocorrelation patterns primarily centred around zero, indicating no significant correlation between positive and negative.

* Daily Average Sentiment (fig. 24)

**A screenshot of a graph

Description automatically generated with low confidence**

(fig. 24)

The first subplot titled "Daily Average Positive Sentiment" depicts the daily average positive sentiment values, with the red dashed line representing the mean positive sentiment value, indicated by the label "Mean.". the highest tweets (1.0) were in December 2017, April 2018, and July 2021

the mean sentiment value is 0.12 which suggests that, on average, the sentiment expressed in the data points classified as positive leans towards a positive or favourable sentiment.

In the second subplot titled "Daily Average Negative Sentiment" illustrates the daily average negative sentiment values, with a red dashed line representing the mean negative sentiment value. The highest (0.5) were in January and April of 2018.

the mean sentiment value is 0.08 indicating that, on average, the sentiment expressed in the data points classified as negative leans towards a negative or unfavourable sentiment.

Additionally, there are two annotations specific to the second subplot:

The annotation labelled "Start of Decline" (in purple an arrowhead indicating a decline) is positioned at the coordinates specified by the values of 'CreatedAt' and the mean negative sentiment value plus 0.01 and it was from the tweets in December 2017 (fig.30a).

The annotation labelled "Start of Incline" (in purple and an arrowhead indicating an incline) is positioned at the coordinates specified by the 15th value of 'CreatedAt' and the y-value of 0.024, and it was from the tweets in December 2017 (fig. 26).



(fig. 26)

"Start of Decline" and "Start of Incline" are both December 2017, and the tweets are much different, making it difficult to have an explanation for the incline or decline of tweets on that date.

* Daily Deviation Sentiment (fig. 27)

**A picture containing text, screenshot, line, plot

Description automatically generated**

(fig. 27)

From both sentiment plots, we can see high variability in the tweets from 2017 to 2023.

* Positive sentiment: the highest was in September 2019 (0.7)
* Negative sentiment: the highest was in Jan 2018 and April 2018 (0.7)

Using Wordcloud, let’s plot common words (fig. 28) and 10 top words (fig. 29) from my dataset.

**A close-up of words

Description automatically generated with low confidence**

(fig. 28)

**A picture containing text, screenshot, font, diagram

Description automatically generated**

(fig. 29)

1. **Interactive Dashboard**

I’m using Panel library because it’s easy to use and have great data visualization capacities (fig. 30).

A screenshot of a computer screen

Description automatically generated with low confidence

**A screenshot of a graph

Description automatically generated with low confidence**A screenshot of a graph

Description automatically generated with low confidence

(fig. 30)

1. **benchmarking tool**

I couldn’t install YCSB, so I decided to use python with the same metrics and connections for Couchbase (fig. 31), MongoDB (fig. 32)., and combined (fig. 33).

**A screenshot of a computer program

Description automatically generated with low confidence**

**A picture containing screenshot, text, line, plot

Description automatically generated**

(fig. 31)

**A screenshot of a computer

Description automatically generated with medium confidence**

**A picture containing text, screenshot, line, plot

Description automatically generated**

(fig. 32)

**A picture containing text, screenshot, font

Description automatically generated**

**A picture containing text, screenshot, software, display

Description automatically generated**

(fig. 33)

Let's analyse the values and compare their performance:

* Run-time (ms):

The runtime indicates the total duration of the load test execution for each database. MongoDB completed the test in 4461.756 milliseconds, which is shorter than Couchbase's 5188.259 milliseconds. A shorter runtime suggests faster processing and potentially better performance.

* Throughput (ops/sec):

Throughput quantifies the quantity of operations or requests completed within a second. MongoDB outperformed Couchbase in terms of throughput, achieving 1.113687 operations per second compared to Couchbase's 0.957739 operations per second. A higher throughput typically signifies superior performance in managing incoming requests.

* Insert Operations:

Both Couchbase and MongoDB executed the same number of insert operations during the load test.

* Average Write Latency (us):

Average write latency measures the average time taken for write operations to complete. MongoDB exhibited a lower average write latency of 0.890128 microseconds compared to Couchbase's 1.035663 microseconds. A lower average latency suggests faster write operations and potentially better performance.

* Min Write Latency (us):

Both Couchbase and MongoDB recorded a minimum write latency of 0.000000 microseconds, indicating that some write operations were extremely fast.

* Max Write Latency (us):

The maximum write latency refers to the longest duration required for a write operation to finish. Couchbase exhibited a higher maximum write latency of 8.683000 microseconds, whereas MongoDB showcased a lower maximum write latency of 2.534000 microseconds. A decreased maximum latency implies superior performance in managing exceptional write operations.95th Percentile Latency (us):

* 95th Percentile Latency (us):

The 95th percentile latency represents the value below which 95% of the observed latencies fall. MongoDB exhibited a lower 95th percentile latency of 1.462200 microseconds compared to Couchbase's 2.000000 microseconds. A lower 95th percentile latency indicates faster performance for most write operations.

* Error Rate:

Couchbase had an error rate of 50%, suggesting that half of the operations during the load test resulted in errors. On the other hand, MongoDB had an error rate of 0%, indicating a more stable performance without errors.

**Conclusion**

The analysis conducted on the dataset of favourited tweets from November 2017 to January 2023 provided valuable insights into sentiment and time series patterns.

For Sentiment Analysis**,** the sentiment distribution appeared to be relatively balanced, with a similar density of tweets for negative, positive, and neutral sentiments. Most tweets were classified as neutral, followed by approximately equal proportions of positive and negative tweets.

Vader Sentiment Analysis helped quantify the sentiment polarity of the tweets, providing sentiment scores ranging from -1 (most negative) to 1 (most positive). Statistical measures for each sentiment partition revealed that Partition 2 had the highest mean values and standard deviations for both positive and negative sentiments, indicating a more diverse range of sentiments within that partition.

For time Series Analysis, Seasonal decomposition of the sentiment data did not reveal any significant seasonal patterns, indicating the absence of recurring sentiment patterns over time. The time series data exhibited a non-stationary nature due to the presence of a trend.

Autocorrelation analysis indicated no significant correlations between positive and negative sentiments at different time lags. Daily average sentiment plots showed varying sentiment patterns over time, with peaks and declines observed at different periods. The mean positive sentiment leaned towards a positive sentiment, while the mean negative sentiment leaned towards a negative sentiment. Daily deviation sentiment plots highlighted high variability in sentiment values across different periods.

The analysis of sentiment and time series patterns in the tweet’s dataset provided insights into the distribution of sentiment, sentiment trends over time, and the presence of variability. These findings can be useful for understanding public sentiment dynamics and for further research and forecasting in sentiment and time series analysis.

For Benchmark tool and based on the load test results, MongoDB outperformed Couchbase in several aspects. MongoDB exhibited a shorter runtime, higher throughput, lower average write latency, lower maximum write latency, lower 95th percentile latency, and no errors during the test. These results suggest that MongoDB performed better than Couchbase in terms of overall performance and stability under the given load test conditions.

**References:**

Banks, M. (2014). MongoDB in Action (2nd ed.). Manning Publications.

Chodorow, K., & Dirolf, M. (2013). MongoDB: The Definitive Guide (2nd ed.). O'Reilly Media.

Copeland, M. (2010). MongoDB: The Definitive Guide. O'Reilly Media.

MongoDB. (n.d.). MongoDB Ecosystem; Retrieved from https://www.mongodb.com/ecosystem.

Rashid, A. (2013). Learning MongoDB. Packt Publishing.

Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice (2nd ed.). OTexts. Retrieved from https://otexts.com/fpp2/