CCT College Dublin

Assessment Cover Page

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

|  |  |
| --- | --- |
| Module Title: | Big Data Process and Storage & Advanced Data Analytics |
| Assessment Title: | **Tweets Sentiment Analysis Prediction and Data Manipulation With BigData Technologies** |
| Lecturer Name: | David McQuaid  Muhammad Iqbal |
| Student Full Name: | Luis Joaquin Morelos Tovar |
| Student Number: | 2023221 |
| Assessment Due Date: | 19 May 2024 |
| Date of Submission: | 22 April 2024 |

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.

# **Introduction**

Social media is a big deal nowadays, and with the rapid grow of existing services and new ones emerging, Big Data manipulation is more necessary than ever. One of this service is Tweeter and it certainly is one of the biggest ones. According to the omnicoreagency website, there are about 500 million tweets sent every day (Aslam, 2024), the dataset used comprises only 1.6 million tweets and it looks already big, 500 million is an incredible big number and an amazing amount of data per day, and that is only for tweeter, there are many other services producing enormous amount of data.

How we handle, process, and storage data is just as important as to what we do with it, meaning, that data by itself is useless, it became something when we are able to extract useful information from it.

So, in this assignment, a dataset containing 1.6 million tweets was used, and to show and how big data technologies can help, a store-read-process-store architecture was design using Big Data technologies such as Hadoop, HBase, etc.

Different technologies can provide different solutions when working with big data, some of them might be faster, or more efficient than others, in this assignment a comparison of an SQL and a NoSQL (MongoDB) databases was done is presented further in this work.

After the data manipulation, an analysis of the dataset using time series was done, this analysis shows the activity of the tweets throughout a period of approximately 3 months and the general sentiment of the tweets contained in the dataset. Using this analysis, a prediction of the sentiment was done and finally shown in an interactive plot.

This work is structured in 3 parts, first **Big Data manipulation, process, storage**, followed by a **Sentiment Analysis & Sentiment Prediction** and finally the **conclusions** drown up from this work.

# **Big Data**

In this work a dataset containing 1.6 million tweets with a size of 219MB was used, the intent is to simulate how does Big Data work in real life. That said, in real life working with large datasets can be incredible hard or complicated of using only one computer, but luckily there are numerously technologies that we can use to share, obtain, process, analyze and store this big data, one of these technologies is Hadoop. Apache Hadoop is an open-source framework that supports processing of large datasets. It can store a large volume of structured, semi-structured and unstructured data in a distributed file system and process them in parallel (Balamurugan Balusamy, 2021) with the help of Hadoop big data can be easily distributed, process and storage it uses nodes to create clusters in order to improve performance and stability for the data.

The tools given in class and the examples set were the basis to complete this ask and the work done throughout this assignment is detailed next.

## **Data Preparation**

In order to work with Big Data a distributed file system is usually used, in this case Hadoop was used to address the issue, a file can be storage in Hadoop and the download in a different location, a simple line using the command *put* helped to move the file from my local disk to the Hadoop system. No further data preparation was needed before storage it in Hadoop and processing it in a MapReduce/Pyspark environment. The activities of the data processing are detailed in the next section

## **Discussion Data Processing and Storage (MongoDB & PySpark)**

Once it was in the Hadoop system it was read in Pyspark . Pyspark was chosen because is an user friendly interface for Apache Spark in Python, this allows us to Pyspark shell to read and analyze data in a distributed system working in an SQL context.

After reading the file from Hadoop we can put it directly in a different database, this could be either in an SQL or a NoSQL database. Each of them has their own pros and cons. According to the previous lectures, HBase has shown to be the fastest technology when compared to others like Hive, MongoBD, Pig, etc. In this work HBase was firstly attempted to use to write-in and read-from the dataset but since it doesn’t have a direct connection to Pyspark the reading part resulted in a failure, for that reason as a second option MongoDB was selected, the reason being, MongoDB allows to handle in a very simple way semi-structured data, and has a direct and simple connector to Pyspark, this made the job or writing and reading much simpler than HBase, and even when is not as fast as HBase is still faster than other technologies such as MySQL.

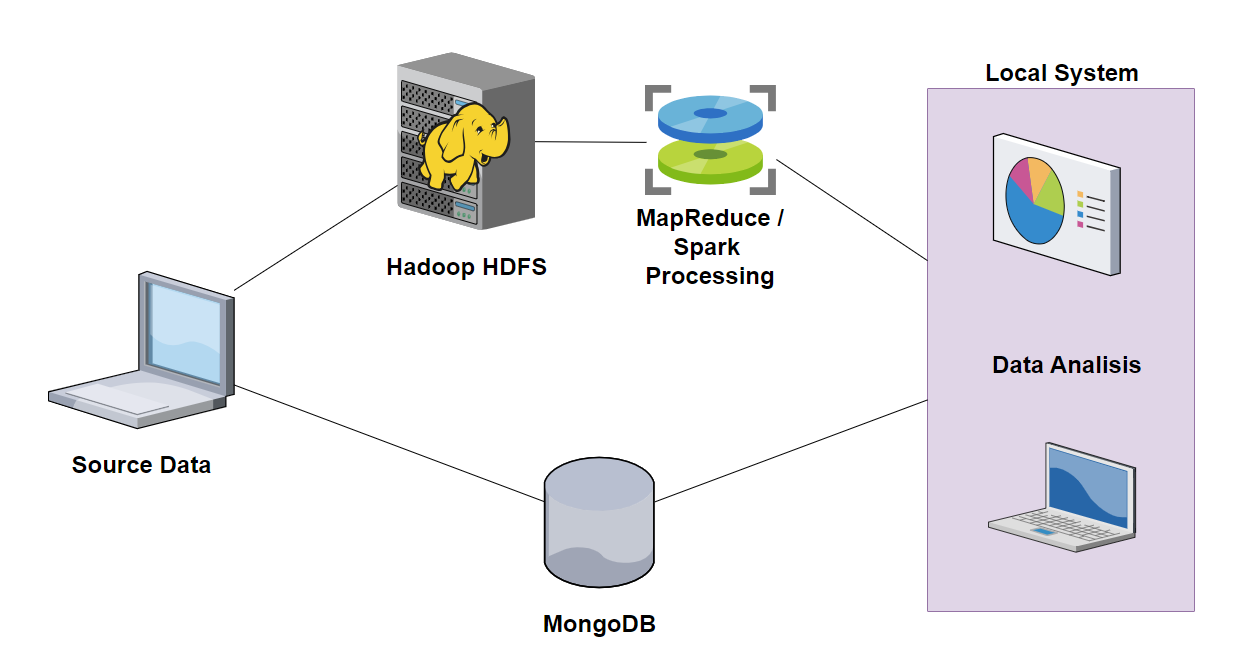
Now that the database is created in MongoDB, it was read with Pyspark, the reason of using Pyspark has already been explain in a previous paragraph, and once read, a primary data cleaning was performed. Deep Learning and analysis are very hardware demanding and due to the limitations of hardware at the moment (intel core i5-4200U, 16GB RAM, 1TB SSD) in this work Pyspark was used only for the data cleaning process.

The whole process of data reading, MongoDB Writing-Reding, data cleaning using Pyspark, and processed data store can be found in the notebook under the name Read-Write-MongoDB in the GitHub link provided at the end of this document.

So, to sum up, Pyspark was used to get the file from Hadoop, feed the database in MongoDB, get the database from MongoDB, afterwards clean the tweets contained in the dataset to prepare the data for the upcoming analysis and finally store the clean dataset as a a csv file in the local disk and create a copy in the Hadoop Distributed File System.

This section describes how the data was prepared and process using a MapReduce/Spark environment (In this case Spark was chosen due to its friendly and intuitive interface <PySpark>) and discusses the rationale and justifications of have chosen MongoDB as the storage system over HBase which has a better performance and was the first tried option and why PySpark whit its friendly interface was chosen over other technologies.

## **Data Architecture**



**Fig. 1 Data Architecture**

Fig. 1 shows the architecture use to build the system in this work, There are two branches after the source data, first one is to upload the data to the Distributed File System Hadoop, once in Hadoop it can be processed using MapReduce or Spark, once processed it can be downloaded in a Local System for its analysis, in this case PySpark was used to analyze the data due to its friendly interface and wide variety of SQL libraries which made it easier to process the data, this technology offers a friendly interface using the Apache Spark Technology.

The second branch is using MongoDB, in which case, the data will be uploaded to MongoDB from the original source and then downloaded into a local system to process the data, once it has been processed it can be uploaded again into MongoDB so it can be analyzed in the source location or the local location.

In this work the MongoDB database was uploaded and read from PySpark just for the sheer purpose of understanding and demonstrate how it works, but this is not necessarily, in a real scenario Hadoop and MongoDB can be used in a separately way.

The advantage that Hadoop offers is that the process can be carry out in the hdfs using Spark or a similar technology whilst MongoDB doesn’t offers the processing tool and it has to be processed with an external tool

## **YCSB Comparative analysis**

For the purpose of testing how SQL and NoSQL technologies perform, a comparison between SQL and MongoDB(NoSQL Technology) using YCSB was carried out, the 3 datasets performing 10,000 operations but at a different read update proportion rate. The results of the analysis are described next:

### **Cleanup Operations**

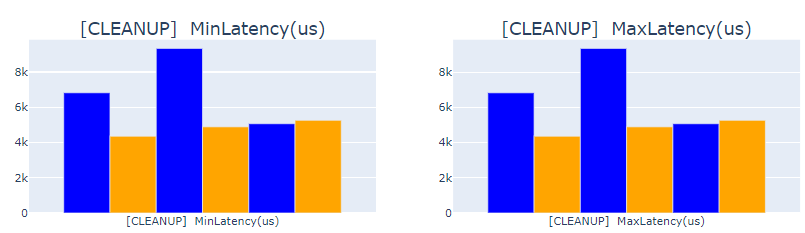


Fig. 2 Clean up Times MySQL & MongoDB

It can be observed in Fig. 2 the difference in cleanup operations with each technology, and its clear how in general terms MongoDB has a better performance in this section, this may be due to MongoDB combining a flexible data model with efficient resource management. It can also be attributed to the fact that cleanup operations in MySQL may involve more steps to ensure data integrity and consistency leading to longer times in cleaning operations.

### **Insert Operations**

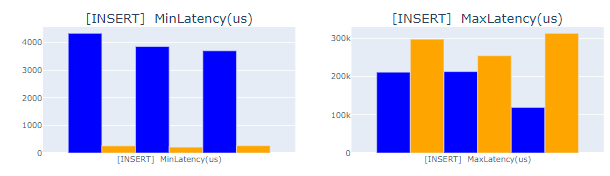


Fig. 3(a) Insert Times MySQL & MongoDB

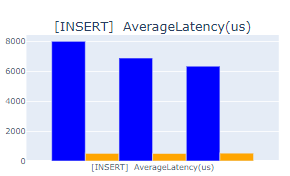


Fig. 3(b) Avrg Insert times MySQL & MongoDB

Similarly, to the clean operations, insert operations can be observed in Fig. 3, in this figure it can be see how MongoDB has the shortest times but also the longest ones, this may be due to its same flexible model but just as it can be observed in fig. 3(b) the average time is still shorter in MongoDB than it is in MySQL, so MongoDB would still be the best choice.

### **99th Percentile Operations**

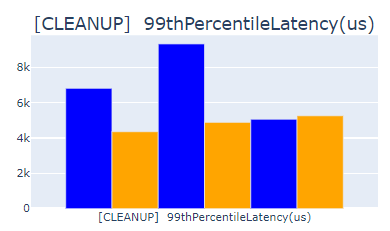
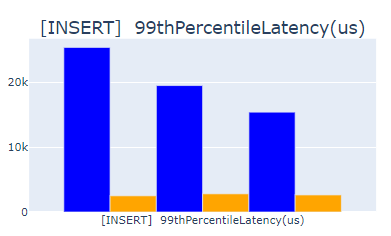
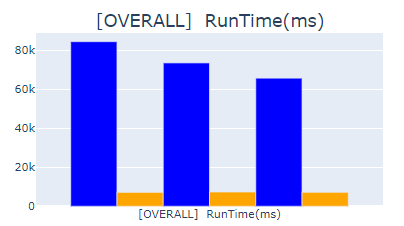
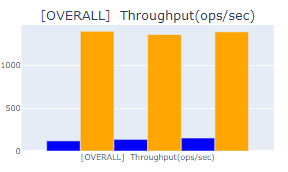
 

Fig. 4 Cleanup-Insert Latency MySQL & MongoDB

It was observed in the previous figures how clean un and insert operations can have different performance, Fig. 4 shows the performance for the INSERT process, MongoDB shows longer times processing some operations, but in Fig. 3 it can be observed that the 99th percent of operations were perfomrmed in less than ¼ of the time it took to MySQL, meaning showing the better performance of MongoDB.

### **Overall operations**

**Fig. 5 Overall RunTime & Trhoughput Operations MySQL & MongoDB**

After analyzing the different metrics for both technologies, it can be observed in Fig. 5 how in an overall perspective MongoDB shows shorter runtimes and higher operations per second, this can be interpreted as a better performance in these technologies, however is important not to forget that selecting the proper technology will depend on the nature of the data and to task to complete.

# **Advanced Data Analytics**

## **Discussion**

### **EDA**

The dataset given was filled with a collection of tweets with no relation between them at all, in this scenario EDA played a major importance because it was the only possible way to tell what was needed to do in order to perform an adequate analysis, so, after displaying the dataset to have an overview and get a general idea of the data in it and since a time series analysis was requested, the first thing to do was check the dataset looking for null values thus a time series analysis cannot be carry out with lost timesteps.

Once the null values were 0, check for duplicates and eliminate them from the dataset and now that there were no spaces in the dataset and no duplicates, a cleaning was performed, first explored the columns, after identifying the relevant and irrelevant columns, the irrelevant ones were dropped to make the analysis easier.

Once the irrelevant columns were removed, it was time to fix the date column, date column originally was in a format containing Day of Week, Month, Day of Month, Hour, Minutes, Seconds, Time Zone and Year, but all of this information was unnecessary not to mention that probably would have mess with the time series sorting, so Date column was split into 6 columns, each of it containing a part of the original one, then renamed the columns to identify them better, afterwards change months names to their equivalent in number, and finally compile a new Date column combining Year, Month, Day & Time, in this new column month needs to be a number so it can be identify as a date column, this will be necessary further in the analysis. A new column mapping Day of Week to its numerical value was added to make visualizations easier to plot in the next stages and finally after Date column cleaning and adding the necessary columns, the old and irrelevant columns were eliminated.

After the Date column cleaning, a check for random tweets with the command *.sample()* was done, the purpose of this was to get an idea of what was In the tweets, at this point some tweets in different languages and with many special characters and even some of them containing only special characters were found, this is because some tweets contain emojis, other languages letters, special characters or user mentions and these when loaded to python are decoded and turn into meaningless data, but getting rid of all of them just because is not rationalizing and probably would have rested a lot of valuable data to this analysis. Back in 2009 tweeter used to allowed a maximum of 140 characters per tweet, this was very helpful because it set a boundary of what could be called a tweet and what was irrelevant and has no importance for the analysis.

So, a first cleaning was performed eliminating websites and the @ in the mentions, so it will decrease the number of characters in the tweets and after this use a simple for loop to count how many tweets were still showing more than 140 characters, then check for the tweets with more than 140 characters and identify what was increasing the numbers of characters, it was identify that, apart from the special characters, words like *quot* or *amp* and were present in the tweets, probably the translation of some special characters in them, but those were nonsense words and were increasing the character count, so a second deep cleaning removing every special character and the identified words was carried out and a new count was performed, showing a massive decrease in the number of tweets, this can bee observed in Fig. 6 below this paragraph.

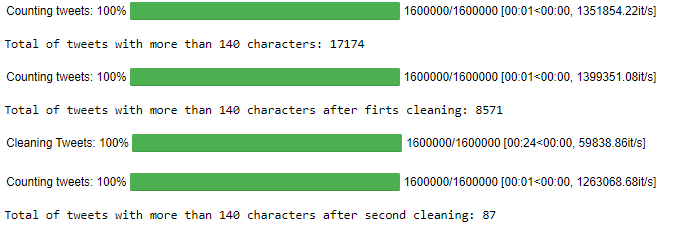


Fig. 5 Tweet cleaning & count

With the cleaning tweets a new column with the length of each tweet was added in order to display a distribution boxplot (Fig. 6), this shows how the character count is distributed and it can be observed that most of the tweets are within the range of 140 and the ones that are not are have over 170 characters, which is a lot.

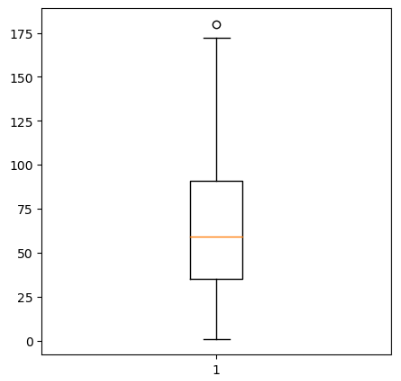


Fig. 6 Number of Characters Boxplot

So after a quick check of what was in the tweets it was observed that no text but special characters and random words were in them (Fig. 7a), so they wouldn’t be able to analyze, on the other hand dropping random rows can mess with the time series, so instead of dropping and for the sake of a smooth distribution, the text was replace by random values of *no*, *neu* and *yes*. The mentioned above can be observed in Fig. 7b

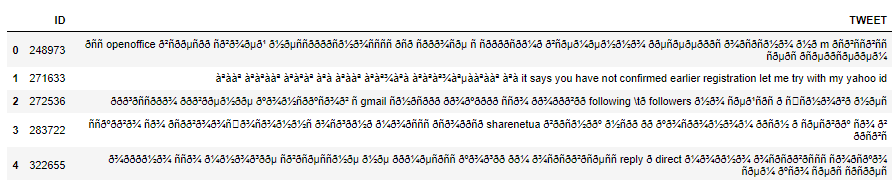


Fig. 7(a) XL Tweets (over 140 characters)

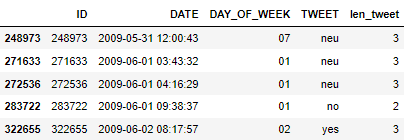


Fig. 7(b) XL Tweet after replace text

The reason for assigning random values and not eliminating them is to have a better distribution of sentiment when analyzing the sentiment and displaying the dataset.

An attempt to translate every tweet was also done but the required time to finish the process was too long so it was decided no to continue with the translation and move forward to the sentient analysis

### **SENTIMENT ANALYSIS**

After the cleaning was done a Sentiment Analysis was performed over the whole dataset, two model were tried, first option was to use RoBerta model which is a heavily pretrained model and has proved to have a better performance when analyzing natural language (Yinhan Liu, 2019) but the limitations on hardware turned it into long processing times (Fig. 8) and delaying the whole analysis so it was change to VADER. VADER was chosen over TextBlob because it shows better results in previous works carried out in this course, so no second thoughts were given when choosing the model.



Fig. 8 RoBerta processing time

An approach using Stop Words module was also tried but it showed no differences when analysing the sentiment of the dataset, so it was decided not to use de module and apply the function directly to the clean dataset. The findings can be observed in the next figure (Fig. 9)

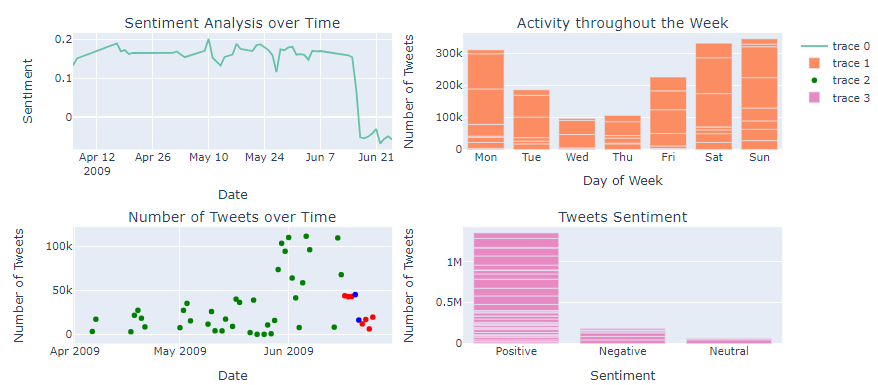


Fig. 9 Sentiment Analysis Results

It can be observed in the top left plot that most of the tweets are positives but there is no trend or seasonality implied, that means that people don’t tweet in a positive, neutral or negative way depending on the date, sentiment can be very random during the year.

In the top right image, it can be observed that there is a trend to use twitter during weekends and Mondays over mid-week days, however this only shows the amount of tweeters posted on every day, but there’s no relation between that and the sentiment on the tweets.

Bottom left plot shows how many tweets were posted in the period of the dataset, it shows more activity during the last part of June, nevertheless, there is not enough data to assume that can be a trend or seasonality activity.

Last image shows hoe many tweets posted showed a positive, netural or negative sentiment, and it can be observed that the majority of the tweets presented a positive sentiment.

### **Prediction Models (AR, ARIMA, RNN)**

Once reached this stage it was necessary to do a little bit more of work on the dataset, the original dataset has no constant time steps, and it can be told by a quick sight to the dataset, what was found is that there were many days missing and this is no good for a time series analysis, so it was decided to implement a piece of code to fill-up the missing days, and random values of sentiment were added to this days, all this in order to be able to performed proper time analysis and time prediction. After having filled the missing days in the time period, a plot was done to compare how the sentiment distribution looked along the time period, and as it was expected a completely different shape of the graph was observed (Fig. 10).

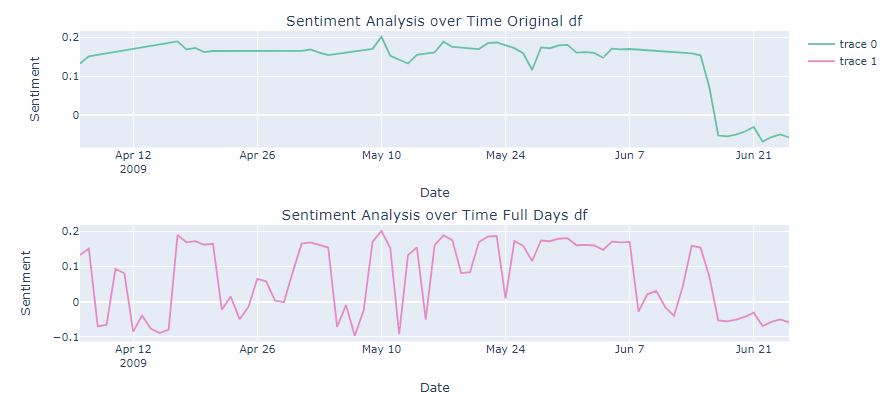


Fig. 10 Sentiment Over Time Original and Date-Filled data frames

AR Model

AR model is the simplest model to apply, it uses previous values to predict future values assuming that a certain relation between the past and future values exists. In this dataset there is no reason to assume that past events don’t impact future ones, so AR model was tried as a first model. AR models are widely used for time series predictions, however this cant be used with non-stationary time series

ARIMA Model

Unlike AR model, ARIMA model can be used with non-stationary time series, and it also can capture trends and seasonal patterns in the time series, this time series didn’t show any seasonal patter or trend (apart from more tweets being posted in weekends) but sometimes there are too much that misses the human eye, and since this is a Data Analysis, it was decided to performed a test using ARIMA Model

RNN

RNN are the most widely used neural networks for time series prediction due to its capability of capture temporal dependencies and when using LSTM it offers enhanced memory capabilities and an impressive ability to selectively remember o forget information over time. This makes RNN the best option when choosing a Neural Network for time series forecasting

## **Hyperparameters**

Hyperparameter was peformed by several trials, however no automatization algorithm or library was used, instead it was testes by trial and error.

**AR MODEL**

For an AR model there is no much to tune when it comes to hyperparameters, first an ACF and PACF plot was done to identify how the lags to be used in the model, and it was tested only with 1 & 2 lags, but the results didn’t show much differences so 1 was chosen

**ARIMA MODEL**

Similarly, to AR Model, for ARIMA model the procedure was to look at the ACF/PACF plots and decide how many lags were to be used, 3 lags are shown in the plot so a trail was made with 2, 7 and 12 lags, in this case the chosen parameters according to the course lectures should have been 2,0,0, but this selection showed an unexpected prediction when testing the predictions, instead of that selection, the final selection was 2,2,1, and these values were achieved by repeated trials ran.

This model showed the highest accuracy compared to the test predictions

**RNN MODEL**

For the RNN different numbers of input neurons, hidden layers, epochs and batch sizes were tested, but none of them showed a major change, when predicting values, so it was kept simple, with 50 input neurons, a hidden layer with 100 neurons, and output layer of 1 neuron, 25 epochs and a batch size of 16.

## **SENTIMEN CHANGE ANALYSIS**

The change of sentiment in the predicted period of time could be due to the lack of relation between the tweets and the amount of them posted every they, it could also be due to where the tweets come from and what is happening at the place where people is tweeting at the moment, we can’t expect people living the war in Ukraine would post the same as people enjoying an evening in Paris, the time would be vey similar, but the sentiment will be definitely different. So this is a tricky part and should be addressed very carefully keeping in mind that the sentiment will change according to the place and time, also setting parameters when collecting the tweets is very important, if the collection is from a variety of sources, languages, locations and times, as it is this one, there’s not much of an analysis that can be really done, it doesn’t matter if we forecast, 1 day, 3 days, 7 days or more.

## **TUFTS PRINCIPLES DAHBOARD DISCUSSION**

The dashboard presents predictions of the selected models based on the dates, it shows the scales of the X and Y axes proportional to the dates and sentiment values respectively and the values presented are directly proportional to the numerical quantities measured. This complies with the first principle

Each prediction model and predicted day is labelled in the dashboard, allowing users to accurate identify and select any specific model and predicted day. Also, further information can be found moving the cursor over the graph in a hover box. This complies with the second principle.

In the presented dashboard, the user can navigate through the different prediction models and predicted days by checking or unchecking boxes, the change in the graphic is mainly of data, there are no significant changes to the shape of the data, but changes in the data displayed. This complies with the third principle.

Fourth principle doesn’t apply to this graph since no money units are being used, however, as mentioned in the first principle, the appropriate units are shown on both axes.

The dashboard presents 2 variables, dates and predicted values, both of them assigned to an axis, and not exceeding the informative dimensions. This complies with the fifth principle.

Last, all data shown in the dashboard is presented within the context of the time period, prediction model or predicted day, there is no loose data without context. This complies with the sixth principle.

Overall, the dashboard displayed in Fig. 10 intends to complies with the 6 principles Tufts presents for a clear and easy understanding of graphical data

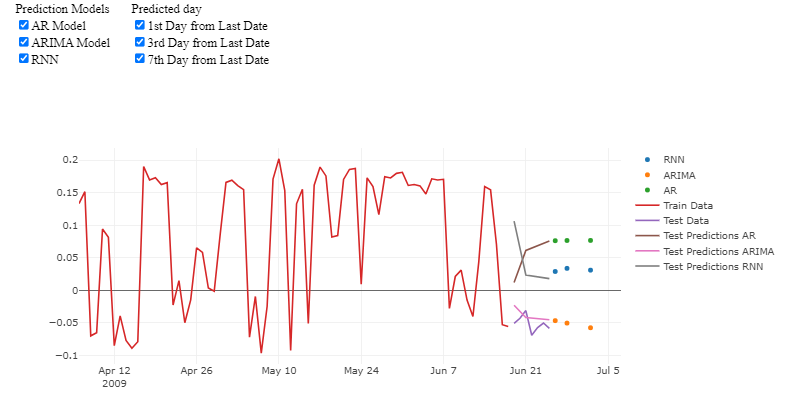


Fig. 10 Prediction Models & Predicted Days Dashboard

# **References**

Aslam, S., 2024. *Omnicore.* [En línea]   
Available at: https://www.omnicoreagency.com/twitter-statistics/  
[Último acceso: 21 May 2024].

Balamurugan Balusamy, N. A. R. S. K. A. H. G., 2021. *Big Data Concepts, Technology and architecture.* First ed. Hoboken: John Wiley & Sons Inc..

Yinhan Liu, e. a., 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach,* Seattle, WA: † Paul G. Allen School of Computer Science & Engineering.