**CCT College Dublin**

**Assessment Cover Page**

|  |  |
| --- | --- |
| **Module Title:** | Advanced Data Analytics  Big Data Storage and Processing |
| **Assessment Title:** | Assessment Two |
| **Lecturer Name:** | David McQuaid  Muhammad Iqbal |
| **Student Full Name:** | Kristine Fae Surat |
| **Student Number:** | sba22208 |
| **Assessment Due Date:** | 26/05/2023 |
| **Date of Submission:** | 24/05/2023 |

**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. |

**Introduction**

Food prices is continuously increasing currently, and I want to get an idea what people are tweeting with a subject topic of “farmers”. This semester, we have learned about the different databases for data handling and storage. The data storage of the dataset used for this study will be discussed on the next section, together with comparisons of NoSQL databases.

In addition to that, it will be described on the next sections the sentimental analysis of tweets. The result of the sentimental analysis will be used for time series forecasting, and tweets in 1 week, 1month and 3 months will be predicted using SARIMA model.

**Data Preparation**

The dataset was gathered from *https://archive.org/details/twitterstream?sort=-publicdate.* It was very challenging for me to gather data with the topic I want to focus on because online sources are very limited and a serious reason was because my laptop kept freezing every time download process starts. Additional tweets were gathered using Twitter API, but since the maximum number of tweets that could be collected was only 100 at a time, I needed to gather more tweets after few days and weeks. I have saved tweets to .json file and tried converting that to .cvs file using Python but the conversion process was not successful due to error warnings, so I just copied the tweets data section by section into an excel spreadsheet. Data collection is the most challenging for me. I believe this is the area I need to improve on and I need to find better techniques to have more efficient data collection.

For my data preparation, I have used MongoDB to store my data. It was particularly challenging for me to use the Virtual Machine for the reason of limited storage, consequently, downloads and data processing were slow. When I was using the Virtual Machine, I preferred to use Spark because it allowed me to use Python Jupyter notebook readily instead of doing the data manipulation on the VM terminal, however, because of the data storage problem the process of data analysis was very slow. I instead chose to download MongoDB on my windows, used that for data storage, and worked with Python through my windows desktop. The dataset was then read into Python Jupyter notebook. Unnecessary columns were dropped until only the Date, Tweeter ID, and Tweets were left.

For my sentiment analysis, I have added additional columns for word count, character count, number of stop words, hashtags, numeric, and letters in uppercases. I wanted to transform uppercases to lowercases to avoid having multiple copies of the same words. I also removed special characters in the tweets. I listed the stop words so I could remove them, as well as the common words. These were removed as their presence will not of any classification of my text data. The rare words were also removed because association between these words and other words is just dominated by noise. In addition to that, rare words were removed, and spellings were corrected. Tokenization, stemming, and lemmatization were applied in preparation for sentiment analysis. Tokenization is a process of dividing the text into a sequence of words or sentences. Stemming was for removal of suffices, like “ing”, “ly”, “s”, etc. by a simple rule-based approach. For this purpose, I used PorterStemmer from the NLTK library. Lemmatization converted the word into its root word, rather than just stripping the suffices. It made use of the vocabulary and did a morphological analysis to obtain the root words. Bigrams were extracted from the reviews using the ngrams function of the textblob. I wanted to capture the language structure, like what letter or word is likely to follow the given one. I worked with bigrams in my Notebook to capture general knowledge.

**Database Comparison**

In Virtual Machine, I tried using MongoDB and HBase as my databases. Personally, I found HBase as the most challenging database among the three due to reasons of getting errors. Figure 25 and Figure 26 in the Appendix section show screenshots of the error warning I got while using the HBase. MongoDB was easier for me to use with the application of MapReduce. Figure 27 and Figure 28 are the screenshots of the Virtual Machine using MapReduce.

MongoDB stored data as Json or Json-like documents. This database is very flexible and responsive in the type or records that can be stored. The documents in this database provide support for a quicker query via indexes. This database reduces the input- output overload that are generally associated with database systems. MongoDB has a feature of high availability and horizontal scalability. Replica sets are included which boast features like data redundancy and automatic failover. MongoDB allows database fine-tuning based on the workload it is serving. It includes incremental operations, indexable array attributes and nested object structure. The disadvantage, however, is that MapReduce remains a slow process. MongoDB suffers from memory hog issues when the database stars to scale. (logz.io)

HBase allows database to store large datasets and also provides analysis in a short period. The biggest advantage of this database is that failover support includes automatic recovery. The read and write of this database adheres to immediate consistency. The disadvantage of HBase is it uses master-slave architecture which demonstrates to be a single point of failure. Failing from one HMaster to another can take time which makes this database to be a not better choice. HBase scales best by adding DataNodes to its cluster, however, it has some high awareness requirements. It is very dependent on HDFS which required five DaraNodes and one NameNode as a minimum. That was the most likely reason why I was getting errors when I was running HBase on my Virtual Machine. Because of its dependency to other systems like Hadoop Distributed Files System (HDFS) and Apache for status management and metadata, designing solution architecture might become complex. (logz.io)

The two databases HBase and MongoDB were used in Virtual Machine by making use of Yahoo Cloud Serving Benchmark (YCSB). Because I failed to compare the two databases on my own work using the Virtual Machine due to the complexity of the system and memory overload, I just reviewed a related article to compare these two databases.

Prasant Vishwakarma and his team conducted a study in 2018 to compare performance analysis of MongoDB and HBase on YSCB. They used most recent versions of the two databases tested in YSCB. They have utilised Testharness script for handling unique test cases which was used to keep regular checks of program behaviour and outputs, and compare the performance of the outputs generated. They used Hadoop and HDFS for their HBase. The two databases were compared using two different workload- Workload A and Workload D. They have utilised five different counts on each workload. The output for each workload were performed three times. They have calculated the average of the output as a way of smoothing their result set. They then visualised the calculated output in a graph. They have analysed the comparison of both databases in terms of reliability, scalability and availability.

A picture containing text, receipt, font, number

Description automatically generated

Figure 1. A screenshot of the Read Average vs Average Latency Value table used by Vishwakarma in their study (Workload A).

(Reference Comparative Performance Analysis of Mongo and HBase on YSCB by Prasant Vishwakarma)

A screenshot of a graph

Description automatically generated with medium confidence

Figure 2. A screenshot of Read Operation vs Average Latency graph generated by Vishwakarma in their study. (Workload A)

(Reference Comparative Performance Analysis of Mongo and HBase on YSCB by Prasant Vishwakarma)

It can be seen on Figure 1 and Figure 2 that the number of read operations increased for both database on all operational counts. Looking at the average latency, it can be inferred from their result that MongoDB performed better because the values of average latency were lower as compared to values for HBase. The lower the average latency value is, the faster the performance. Therefore, it can be inferred that the performance of MongoDB is faster than that of HBase. For all operational counts, MongoDB has better and quicker read index performance in comparison to HBase. Figure 3 and Figure 4 show the graphical representation of update operation against average latency that their team produced. Workload A represented that MongoDB had faster update operation as compared to HBase, thus, it can be inferred that MongoDB has better performance.

A picture containing text, screenshot, font, number

Description automatically generated

Figure 3. A screenshot of the Updated Operation vs Updated Average Latency table used by Vishwakarma in their study (Workload A).

(Reference Comparative Performance Analysis of Mongo and HBase on YSCB by Prasant Vishwakarma)

A picture containing text, screenshot, diagram, plot

Description automatically generated

Figure 4. A screenshot of Updated Operation vs Average Latency graph generated by Vishwakarma in their study. (Workload A)

(Reference Comparative Performance Analysis of Mongo and HBase on YSCB by Prasant Vishwakarma)

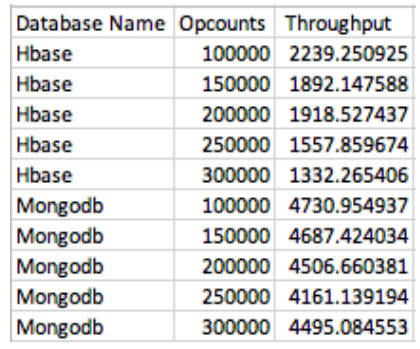


Figure 5. A screenshot of the Operational Counts vs Throughput Value table used by Vishwakarma in their study (Workload A).

(Reference Comparative Performance Analysis of Mongo and HBase on YSCB by Prasant Vishwakarma)

A picture containing text, screenshot, plot, parallel

Description automatically generated

Figure 6. A screenshot of Operational Count vs Throughput graph generated by Vishwakarma in their study.

(Workload A)

(Reference Comparative Performance Analysis of Mongo and HBase on YSCB by Prasant Vishwakarma)

The number of Operations processed per second is called the overall throughput. If the throughput is higher, the database is expected to perform better. Figure 6 summarises that MongoDB has better performance as compared to HBase.

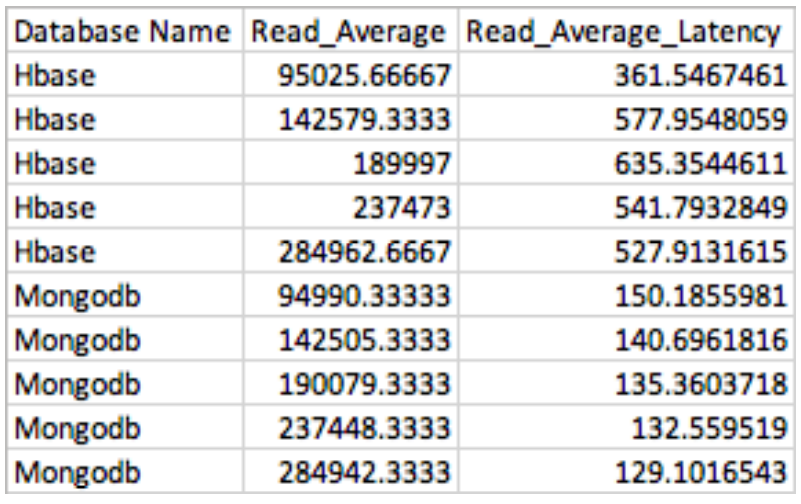


Figure 7. A screenshot of the Read Operation vs Read Average Latency Value table used by Vishwakarma in their study. (Workload D)

(Reference Comparative Performance Analysis of Mongo and HBase on YSCB by Prasant Vishwakarma)

A picture containing text, screenshot, diagram, plot

Description automatically generated

Figure 8. A screenshot of Read Operation vs Average Latency graph generated by Vishwakarma in their study. (Workload D)

(Reference Comparative Performance Analysis of Mongo and HBase on YSCB by Prasant Vishwakarma)

Figure 7 and Figure 8 describe the average latency of read operation for both databases for all five operational counts on Workload D. Same as workload A, average latency of HBase is so much higher than that of MongoDB, hence, performance of HBase is comparatively poor than MongoDB.

A picture containing text, screenshot, font, number

Description automatically generated

Figure 9. A screenshot of Insert Operation vs Average Latency Table generated by Vishwakarma in their study. (Workload D)

(Reference Comparative Performance Analysis of Mongo and HBase on YSCB by Prasant Vishwakarma)

A picture containing text, screenshot, diagram, plot

Description automatically generated

Figure 10. A screenshot of Insert Operation vs Average Latency graph generated by Vishwakarma in their study. (Workload D)

(Reference Comparative Performance Analysis of Mongo and HBase on YSCB by Prasant Vishwakarma)

Figure 9 and Figure 10 are illustrations of insert operation against average latency for Workload D. The figures show that the Average Update Latency of MongoDB are 3 to 4 times less than HBase, thus, MongoDB has better performance than HBase.

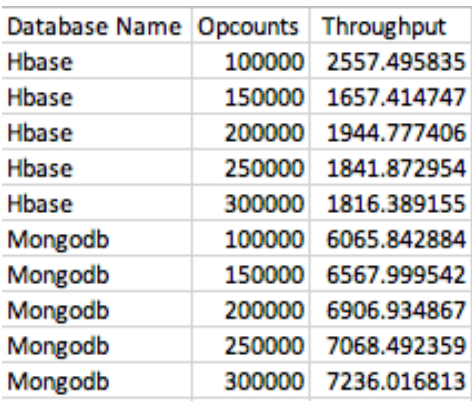


Figure 11. A screenshot of the Operational Counts vs Throughputs Value table used by Vishwakarma in their study (Workload A).

(Reference Comparative Performance Analysis of Mongo and HBase on YSCB by Prasant Vishwakarma)

A picture containing text, screenshot, diagram, plot

Description automatically generated

Figure 12. A screenshot of Operational Counts vs Throughputs graph generated by Vishwakarma in their study. (Workload D)

(Reference Comparative Performance Analysis of Mongo and HBase on YSCB by Prasant Vishwakarma)

Figure 11 and Figure 12 display the operational count and throughputs of Workload D. It can be observed from Figure 12 that the throughputs of HBase are a 3 to 4 times less than throughputs of MongoDB, therefore, MongoDB is considered better than HBase.

In conclusion, the performance of HBase and MongoDB provides measures of availability, scalability and reliability which were tested on YCSB. The two NoSQL databases were compared by Vishwakarma and his team with the considerations of read, insert and overall throughput performance and they have concluded that MongoDB has a better performance than HBase.

**Sentiment Analysis**

After data preparation, sentiment analysis was started by getting the term frequency of tweets. Term frequency is the ratio of the count of a word present in a sentence, to the length of the sentence. Each tweet was given a numeric sentiment rating then converted to polarity rating where zero becomes neutral, less than zero as negative and more than zero as positive.

Figure 13 below shows the visualisation of the sentiments. 800 samples were selected for the analysis because that was the optimum sample number that gave me best accuracy result. It can be observed that most of the tweets were neutral. Negative sentiments were the least among the three.



Figure 13. Sentiment Analysis graph

Train test split were applied in Python in order to have a train and test dataset. CountVectorizer( ) was used for Vectorization. Frequency, Inverse Frequency used TfidTransformer ( ). Different layers were added with sequential model.

A picture containing text, screenshot, font, document

Description automatically generated

Figure 14. Fitting TFID Model in Sentiment Analysis

Figure 14 shows the result of accuracy after fitting the model. It can be seen that the accuracy at epoch 1 was only 52%, then increased to 99.7% after three more epochs. Fitting the model stopped at epoch 4.



Figure 15. Test Accuracy of TFID model

The test accuracy of the model seemed to have 85.52%, thus, TFID is a good model for sentiment analysis.

**Time Series Forecast**

I have saved my sentiment analysis output from the previous section to a .csv file and used for my time series forecasting.

Figure 16 shows the graph of Tweets converted into numerical sentiments from sentiment analysis. It can be observed form the graph that there are more positive sentiments as compared to negative sentiments.

A red lines on a white background

Description automatically generated with low confidence

Figure 16. Tweets Sentiments from May 2020 to May 2023

A picture containing diagram, screenshot, rectangle, line

Description automatically generated

Figure 17. Box plot of Tweets

Figure 17 displays the boxplot of yearly tweets and monthly tweets to summarise the dispersion of the dataset used. It can be observed that there were plenty of outliers that lie outside the overall distribution pattern for both quadrants. Months April and May were showing that the dataset skewed towards the negative sentiments, and the rest of the months skewed to positive sentiments.

A blue line graph on a white background

Description automatically generated with low confidence

Figure 18. Graph of Weekly Tweets

Figure 18 shows the weekly data. It can be observed that the graph is cleaner as compared to Figure 16 and it can still be pictured that there were more positive sentiments than negative.

Auto Regressive Integrated Moving Average (ARIMA) is the model used for time series forecasting because it explains time series and gives forecasts based on the past values including the lags and lagged forecast errors. ARIMA forecasting is like a linear regression. The predictors depends on the parameters: number of auto-regressive terms or lags of dependent variable; number of moving average or lagged forecast errors in prediction; and number of differences.

ARIMA model is used for non-seasonal time series only and does not support seasonality. For this experiment, I want to confirm if my dataset is seasonal. I used auto\_arima() in my prediction which used a stepwise approach to each multiple combinations of parameters and chose the best model that has the least Akaike Information Criterion (AIC). AIC is a mathematical method used for evaluation of models how well it firs the data it was generated from.

The train and test set were split then applied autoarima to decide parameters. The predicted value for test was determined, plot the train, the rest and predicted data, then evaluated the accuracy of the forecast.

A picture containing text, font, line, screenshot

Description automatically generated

Figure 19. Forecasting using ARIMA model

The Mean Absolute Percentage Error I got using ARIMA model was 0.55, which means ARIMA has high accuracy of prediction using my dataset. However, Figure 19 shows that the predictions were constant at zero. This means that my dataset is verified as seasonal and that ARIMA model is not the right model for my prediction. The result from the forecast does not support the ARIMA model. ARIMA forecast do not show season trend data that was why the result of the prediction was constant at zero.

A picture containing text, handwriting, screenshot, font

Description automatically generated

Figure 20. Forecasting using SARIMA model

My time series dataset has defined seasonality, hence, SARIMA model is the right model to use for my forecasting. For this model, seasonal differencing happens where the values are subtracted from the previous season. Mean Absolute Percentage Error I got using the SARIMA model was 0.83. The MAPE is considerably high when SARIMA model is used which means that the accuracy could not support the model. It is even higher than the MAPE I got for the ARIMA model which should not be the case because my data is seasonal, and the MAPE should be decreased for SARIMA. This time I will blame my dataset. A bigger, more consistent data may change the accuracy. It can be observed from Figure 20 that the prediction was not constant at zero in comparison to the prediction in Figure 19.

With that, I used SARIMA model for my sentiments forecast. I wanted to see the predictions in 1 week, 1 month and 3 months.

A screen shot of a computer

Description automatically generated with low confidence

Figure 21. Future Sentiment in 1 week

A picture containing handwriting, text, font, screenshot

Description automatically generated

Figure 22. Future Sentiment in 1 month

A picture containing text, handwriting, font, screenshot

Description automatically generated

Figure 23. Future Sentiment in 3 months

It can be observed from Figure 21 that the prediction was for 2034. The reason behind this could be because the dataset available for forecasting did not provide constant daily sentiments that could give a basis of prediction. The projection I was getting from Python somehow gave 7-day projection for 2034. It can be viewed, however, that the projection of sentiments for 1 week are all positive.

Figure 22 also shows positive sentiments forecast in 1 month. Figure 23 displays prediction in 3 months and it can be observed on the prediction that sentiment will be positive and neutral.

In summary, time series forecast results were telling that the future sentiments about farmers will be positive sentiment if not neutral.

**Conclusion**

Tweets with topic of ‘Farmers’ were gathered using different online sources. The dataset gathered was stored in MondoDB. The tweets data was cleaned and prepared for sentiment analysis. It was observed that most of the tweets from May 2020 to May 2023 were positive sentiments and negative sentiments was the least among the three categories. Time Series forecasting was done for 1 week, 1 month and 3 months. For the forecasting, SARIMA model was used because the dataset is seasonal. The predictions showed that there will be more positive sentiments than neutral on future tweets about farmers, and no negative sentiments. This could mean that there will be no bad comments about farmers in the future.

MongoDB and HBase were compared in a previous study and that paper was reviewed on this paper. It was concluded that the performance of HBase and MongoDB provided measures of availability, scalability and reliability were tested on YCSB. The two NoSQL databases compared with the considerations of read, insert and overall throughput performance and MongoDB had a better performance than HBase.

**Appendix 1:** Screenshots from Oracle Virtual Machine

A screenshot of a computer

Description automatically generated

Figure 24. Screenshot using YSCB

A picture containing text, screenshot, font

Description automatically generated

Figure 25. Screenshot using HBase



Figure 26. Screenshot or error using HBase

A screenshot of a computer program

Description automatically generated with medium confidence

Figure 27. Screenshot using MapReduce- Part 1

A screenshot of a computer screen

Description automatically generated with medium confidence

Figure 28. Screenshot using MapReduce- Part 2

**References**

*Cassandra vs. Mongodb vs. Hbase: A comparison of NoSQL databases* (2021) *Logz.io*. Available at: https://logz.io/blog/nosql-database-comparison/ (Accessed: 24 May 2023).

Pandian, S. (2023) *Time series analysis and forecasting: Data-Driven Insights (updated 2023)*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-to-time-series-analysis/ (Accessed: 24 May 2023).

Prasant Vishwakarma (2018) *Comparative Performance Analysis of MongoDB and HBase on YCSB,* <https://www.researchgate.net/publication/330823015>

Github account: