**INTRODUCTION**

**Background**

Social networking sites connect people in the world, allowing them to share pictures, content, videos and share their first-hand opinions on various issues. Big data analytics techniques are highly applied in the social networks because they are characterized by the 5V (Velocity, Volume, Veracity, Value AND Variety) s of big data.(Bazzaz Abkenar *et al.*, 2021). Some examples of social networking sites include Twitter, Facebook etc. Due to social media providing a big source of data, there has been an increase in application of machine learning, deep learning and time series techniques to better understand various problems. A lot of these analysis has been done mostly on twitter data. Twitter which has over 313 million monthly active users and which in a day over 500 million tweets are made, is one of the most utilised social media platforms when it comes to data.(Jianqiang and Xiaolin, 2017)

Some key techniques employed for understanding social media data include, sentiment analysis, times

**Problem statement**

**DATA**

The dataset used for this project was project Tweets dataset, a csv dataset with 1,600,000 observations and five features extracted using the twitter api. It contains the following 5 fields:

* ids: The id of the tweet (eg. 4587)
* date: the date of the tweet (eg. Sat May 16 23:58:44 UTC 2009)
* flag: The query (eg. lyx). If there is no query, then this value is NO\_QUERY.
* user: the user that tweeted (eg. bobthebuilder)
* text: the text of the tweet (eg. Lyx is cool)

**Big data Processing and Storage**

Big data processing is techniques utilised to access large scale data and extract meaningful information from them for decision making.(Mehdipour, Noori and Javadi, 2016), while big data storage are storage technologies that are not relational database systems that can be able to address the Volume, variety and velocity challenges of data.(Strohbach *et al.*, 2016). There are different big data storage and processing technologies available. Processing technologies include Hadoop Map-Reduce or Apache Spark, etc. While storage include either SQL or NoSQL databases such as HBase, HIVE, Spark SQL, Cassandra, MongoDB. Etc

For this project Apache Spark was used for Processing the Project Tweets Data, while MongoDB and spark SQL were used to Populate, Store and save Processed Data.

**Spark SQL**

This is a spark module for structured data processing.

Following your analysis, you are then required to make a time series forecast of the sentiment **of the entire dataset** at 1 day, 3 days and 7 days going forward. This forecast must be displayed as a dynamic dashboard.

Your project must incorporate the following elements:

* Source dataset(s) can be stored into an appropriate SQL/ NoSQL database(s) prior to processing by MapReduce / Spark (HBase / HIVE / Spark SQL /Cassandra / MongoDB / etc.) The data can be populated into the NoSQL database using an appropriate tool (Hadoop/ Spark etc.)
* Post Map-reduce processing dataset(s) can be stored into an appropriate NoSQL database(s) (Follow a similar choice as in the previous step)
* Store the data and then follow-up analysis on the output data. It can be extracted from the NoSQL database into another format, using an appropriate tool, if necessary (e.g. extract to CSV to import into R/ Python etc.).
* Devise and implement a test strategy in order to perform a comparative analysis of the capabilities of any two databases (MySQL, MongoDB, Cassandra, HBase and CouchDB) in terms of the performance. You should record a set of appropriate metrics and perform a quantitative analysis for comparison purposes between the two chosen database systems.
* Provide evidence and justification of your choice of sentiment extraction techniques.
* Explore at least 2 methods of time-series forecasting including at least 1 Neural Network and 1 autoregressive model (ARIMA, SARIMA etc…) . (Hint: that this is a Short time series, How are you going to handle this?)
* Evidence and justify your choices for your final analysis and include your forecasts at 1 day, 3 days and 7 days going forward.
* Your dashboard must be dynamic and interactive. Include your design rationale expressing Tufts principles.

**Deliverables:**

The results of the analysis must be presented in the form of a project report. This report should discuss the storage and processing of big data using advanced data analytics techniques. The report should be 3000 ± 10% words in length (excluding references, titles, and code) and must follow the Harvard styles format in addition to employing appropriate referencing methods and academic writing style. The report should include the following:

**Big Data**

1. Details of the data storage and processing activities carried out, including preparation of the data and processing the data in a MapReduce/ Spark environment;**[0-30]**
2. Comparative analysis for at least two databases (one SQL and at least one NOSQL) using YCSB.[0**-30]**

**3. YCSB FOR Comparative analysis OF MYSQL AND MONGODB**

Yahoo cloud service benchmark client is an open-source license tool used to benchmark new cloud database systems. Through YCSB one can be able to benchmark multiple systems and compare them by creating “workloads”. (Gaikwad and Goje, 2015). A YCSB Comparative analysis was conducted to compare MySQL and MongoDB.

The comparison involved the following areas: -

1. *Workloads a, b, c*
2. *Read and update options*
3. *Distribution*

The comparison involved comparing the two databases performance based on record counts 1,000, 10,000 and 100,000. Further the read and update proportions were also compared for different proportions and lastly the request distributions compared were Zipfian and uniform distribution

|  |  |  |
| --- | --- | --- |
| **Comparison Parameters** | **MYSQL** | **MONGODB** |
| Record Counts | 1,000, 10,000 and 100,000 | |
| Request distribution | Zipfian and Uniform | |
| Workload A – Update heavy | * Read: 50% and Update: 50% * Read: 70% and Update: 30% | |
| Workload B- Read heavy | * Read: 95% and Update: 5% * Read: 70% and Update: 30/5 | |
| Workload C- Read only | * Read: 100% and Update:0% * Read: 70% and Update: 30% | |

**i.) Comparison of MySQL vs MongoDB Using workload A using Zipfian Distribution (50/50 and 70/30 Read and update).**

From table above, as the record accounts increase for both databases, the runtime also increases. MongoDB has lower runtime than MySQL when running all the three record counts.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **50/50 (**Zipfian) | | | | | | | | | | | |
| **Read/update proportions** | **MySQL** | | | | | **MongoDB** | | | | | |
| **Record Counts** | **1000** | **10,000** | | **100,000** | | **1000** | | **10,000** | | **100,000** | |
| RunTime(ms) | 9957 | 71289 | | 873740 | | 1074 | | 4381 | | 20910 | |
| Throughput(ops/sec) | 100.431857 | 140.274096 | | 114.450523 | | 931.098696 | | 2282.58388 | | 4782.40077 | |
| Average Latency(us) | 7741.987 | 7075.8719 | | 8724.64075 | | 582.738 | | 369.638 | | 195.51202 | |
| Min Latency(us) | 3046 | 3994 | | 3294 | | 170 | | 56 | | 35 | |
| Max Latency(us) | 128319 | 48863 | | 69271551 | | 146047 | | 158591 | | 1950719 | |
| 95thPercentileLatency(us) | 12063 | 10143 | | 13183 | | 1037 | | 796 | | 454 | |
| 99thPercentileLatency(us) | 25023 | 16927 | | 29855 | | 2001 | | 1394 | | 780 | |
| **70/30 (**Zipfian**)** | | | | | | | | | | | |
| **Read/update proportions** | **MySQL** | | | | | **MongoDB** | | | | | |
| **Record Counts** | **1,000** | **10,000** | | **100,000** | | **1,000** | | **10,000** | | **100,000** | |
| RunTime(ms) | 6865 | 67359 | | 706801 | | 1864 | | 7666 | | 42974 | |
| Throughput(ops/sec) | 145.666424 | 148.458261 | | 141.482539 | | 536.480687 | | 1304.46126 | | 2326.98841 | |
| Average Latency(us) | 6386.185 | 6679.5067 | | 7050.07888 | | 708.68 | | 650.1487 | | 412.20452 | |
| Min Latency(us) | 3688 | 3582 | | 2712 | | 157 | | 65 | | 34 | |
| Max Latency(us) | 42591 | 58879 | | 169983 | | 154879 | | 228991 | | 159999 | |
| 95thPercentileLatency(us) | 9599 | 9199 | | 10951 | | 1320 | | 1647 | | 884 | |
| 99thPercentileLatency(us) | 19199 | 15391 | | 14855 | | 3889 | | 5223 | | 2305 | |
| **50/50 (Uniform**) | | | | | | | | | | | |
| **Read/update proportions** | **MySQL** | | | | | | **MongoDB** | | | | | |
| **Record Counts** | **1000** | | **10,000** | | **100,000** | | **1000** | | **10,000** | | **100,000** | |
| RunTime(ms) | 8267 | | 99077 | | 1052529 | | 1756 | | 4860 | | 20853 | |
| Throughput(ops/sec) | 120.9628644 | | 100.9315987 | | 95.00925865 | | 569.476082 | | 2057.613169 | | 4795.473073 | |
| Average Latency(us) | 7723.14 | | 9829.2211 | | 10504.89543 | | 706.401 | | 382.9785 | | 196.65383 | |
| Min Latency(us) | 4528 | | 3524 | | 3384 | | 142 | | 58 | | 35 | |
| Max Latency(us) | 74879 | | 272127 | | 336895 | | 150911 | | 141695 | | 208511 | |
| 95thPercentileLatency(us) | 11439 | | 14167 | | 16639 | | 1348 | | 744 | | 615 | |
| 99thPercentileLatency(us) | 52447 | | 72639 | | 23951 | | 2921 | | 1543 | | 984 | |
| **70/30 (uniform)** | | | | | | | | | | | | |
| **Read/update proportions** | **MySQL** | | | | | | **MongoDB** | | | | | |
| **Record Counts** | **1,000** | | **10,000** | | **100,000** | | **1,000** | | **10,000** | | **100,000** | |
| RunTime(ms) | 13010 | | 75478 | | 866234 | | 1209 | | 3855 | | 19506 | |
| Throughput(ops/sec) | 76.86395081 | | 132.4889372 | | 115.4422477 | | 827.1298594 | | 2594.033722 | | 5126.627704 | |
| Average Latency(us) | 11686.936 | | 7475.8006 | | 8649.02499 | | 489.766 | | 295.5393 | | 182.21613 | |
| Min Latency(us) | 6052 | | 4108 | | 3776 | | 141 | | 44 | | 35 | |
| Max Latency(us) | 106239 | | 133759 | | 365055 | | 142719 | | 231935 | | 280831 | |
| 95thPercentileLatency(us) | 18607 | | 10679 | | 13735 | | 916 | | 647 | | 606 | |
| 99thPercentileLatency(us) | 53951 | | 24127 | | 19855 | | 1481 | | 1475 | | 966 | |

**WORKLOAD B**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **95/5 (**Zipfian) | | | | | | |
| **Read/update proportions** | **MySQL** | | | **MongoDB** | | |
| **Record Counts** | **1000** | **10,000** | **100,000** | **1000** | **10,000** | **100,000** |
| RunTime(ms) | 7103 | 74965 | 691730 | 1392 | 6781 | 20141 |
| Throughput(ops/sec) | 140.785584 | 133.395585 | 144.565076 | 718.3908046 | 1474.708745 | 4964.996773 |
| Average Latency(us) | 6577.019 | 7435.8522 | 6906.23396 | 746.279 | 459.3124 | 189.63952 |
| Min Latency(us) | 4056 | 3086 | 3272 | 211 | 62 | 38 |
| Max Latency(us) | 70783 | 106367 | 120959 | 104703 | 258943 | 636415 |
| 95thPercentileLatency(us) | 9927 | 15135 | 10535 | 1222 | 836 | 615 |
| 99thPercentileLatency(us) | 19551 | 22255 | 14239 | 3413 | 2301 | 1162 |
| **70/30 (**Zipfian**)** | | | | | | |
| **Read/update proportions** | **MySQL** | | | **MongoDB** | | |
| **Record Counts** | **1,000** | **10,000** | **100,000** | **1,000** | **10,000** | **100,000** |
| RunTime(ms) | 7790 | 65809 | 720646 | 1609 | 5328 | 21621 |
| Throughput(ops/sec) | 128.3697047 | 151.9548998 | 138.7643864 | 621.5040398 | 1876.876877 | 4625.132973 |
| Average Latency(us) | 7132.117 | 6507.8252 | 7193.95318 | 708.68 | 650.1487 | 412.20452 |
| Min Latency(us) | 4070 | 3522 | 3118 | 952.147 | 468.6321 | 206.54024 |
| Max Latency(us) | 61887 | 52575 | 117567 | 180 | 66 | 40 |
| 95thPercentileLatency(us) | 11391 | 8783 | 11559 | 198783 | 182399 | 135167 |
| 99thPercentileLatency(us) | 22687 | 15287 | 17359 | 1404 | 852 | 641 |
| **95/5(uniform)** | | | | | | |
| **Read/update proportions** | **MySQL** | | | **MongoDB** | | |
| **Record Counts** | **1,000** | **10,000** | **100,000** | **1,000** | **10,000** | **100,000** |
| RunTime(ms) | 7790 | 65809 | 720646 | 1609 | 5328 | 21621 |
| Throughput(ops/sec) | 128.3697047 | 151.9548998 | 138.7643864 | 621.5040398 | 1876.876877 | 4625.132973 |
| Average Latency(us) | 7132.117 | 6507.8252 | 7193.95318 | 708.68 | 650.1487 | 412.20452 |
| Min Latency(us) | 4070 | 3522 | 3118 | 952.147 | 468.6321 | 206.54024 |
| Max Latency(us) | 61887 | 52575 | 117567 | 180 | 66 | 40 |
| 95thPercentileLatency(us) | 11391 | 8783 | 11559 | 198783 | 182399 | 135167 |
| 99thPercentileLatency(us) | 22687 | 15287 | 17359 | 1404 | 852 | 641 |
| **70/30(uniform)** | | | | | | |
| **Read/update proportions** | **MySQL** | | | **MongoDB** | | |
| **Record Counts** | **1,000** | **10,000** | **100,000** | **1,000** | **10,000** | **100,000** |
| RunTime(ms) | 7790 | 65809 | 720646 | 1609 | 5328 | 21621 |
| Throughput(ops/sec) | 128.3697047 | 151.9548998 | 138.7643864 | 621.5040398 | 1876.876877 | 4625.132973 |
| Average Latency(us) | 7132.117 | 6507.8252 | 7193.95318 | 708.68 | 650.1487 | 412.20452 |
| Min Latency(us) | 4070 | 3522 | 3118 | 952.147 | 468.6321 | 206.54024 |
| Max Latency(us) | 61887 | 52575 | 117567 | 180 | 66 | 40 |
| 95thPercentileLatency(us) | 11391 | 8783 | 11559 | 198783 | 182399 | 135167 |
| 99thPercentileLatency(us) | 22687 | 15287 | 17359 | 1404 | 852 | 641 |

1. A discussion of the rationale and justification for the choices you have made in terms of data processing and storage, programming language choice, that you have implemented.**[0-20]**
2. Design the architecture for the processing of big data using all the necessary technologies (HADOOP/SPARK,NOSQL/SQL databases and programming). Present your Design in the form of a diagram and discussion in your report **.[0-20]**

**Note that MapReduce-style processing in this instance is considered to include platforms such as Apache Spark.**

**Advanced Data Analytics**

1. A discussion of the rationale, evaluation, and justification for the choices you have made in terms of EDA, data wrangling, machine learning models and algorithms that you have implemented**.[0-40]**
2. **E**valuation and justification of the hyperparameter tuning techniques that you have used **[0-20]**
3. Your analysis of any change sentiment that occurs and your forecast of the sentiment at 1 day, 3 days and 7 days going forward**[0-20]**
4. Presentation of results by making appropriate use of figures along with caption, tables, etc and your dashboard for your forecast, Discuss Tufts Principles in relation to your Dashboard **.[0-20]**

**SUBMISSION:**

**Submission Requirements** All assessment submissions must meet the minimum requirements listed below. Failure to do so may have implications for the mark awarded.

All assessment submissions must:

* 3000 words +- 10% (excluding references, titles, citations and quotes)
* Word Document for report (No PDF’s), Jupyter notebook for code, Screencast for practical demonstration.
* Be submitted by the deadline date specified or be subject to late submission penalties
* Be submitted via Moodle upload
* Use [Harvard Referencing](http://40.115.124.2/sp/subjects/guide.php?subject=harvardref) when citing third party material
* Be the student’s own work.
* Include the CCT assessment cover page.

**Additional Information**

* Lecturers are not required to review draft assessment submissions.
* In accordance with CCT policy, feedback to learners may be provided in written, audio or video format and can be provided as individual learner feedback, small group feedback or whole class feedback.
* Results and feedback will only be issued when assessments have been marked and moderated / reviewed by a second examiner.
* Additional feedback may be requested by contacting your lecturer AFTER the publication of results,Additional feedback may be provided as individual, small group or whole class feedback. Lecturers are not obliged to respond to email requests for additional feedback where this is not the specified process or to respond to further requests for feedback following the additional feedback.
* Following receipt of feedback, where a student believes there has been an error in the marks or feedback received, they should avail of the recheck and review process and should not attempt to get a revised mark / feedback by directly approaching the lecturer. Lecturers are not authorised to amend published marks outside of the recheck and review process or the Board of Examiners process.
* Students are advised that disagreement with an academic judgement is not grounds for review.
* For additional support with academic writing and referencing students are advised to contact the CCT Library Service or access the [CCT Learning Space](http://learningspace.cct.ie/subjects/index.php).
* For additional support with subject matter content students are advised to contact the [CCT Student Mentoring Academy](https://moodle.cct.ie/mod/forum/view.php?id=55148)
* For additional support with IT subject content, students are advised to access the [CCT Support Hub](https://moodle.cct.ie/course/view.php?id=1861).

**CCT College Dublin**

**Assessment Cover Page**

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

|  |  |
| --- | --- |
| **Module Title:** |  |
| **Assessment Title:** |  |
| **Lecturer Name:** |  |
| **Student Full Name:** |  |
| **Student Number:** |  |
| **Assessment Due Date:** |  |
| **Date of Submission:** |  |

**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. |