Academic City University College – Accra Ghana
Society for Multidisciplinary & Advanced Research Techniques (SMART) Africa
Tony Blair Institute for Global Change
FAIR Forward – Artificial Intelligence for All - Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) GmbH

Accra Bespoke Multidisciplinary Innovations Conference (BMIC)

& The Africa AI Stakeholders' Summit 14th December, 2021

A Gitbash-Generative Model for an Enhanced Query Processing In a Distributed Database System

Enyindah, Promise, Oghenekaro, Linda Uchenna & Marcus, Chigoziri Bobby.
Department of Computer Science
University of Port Harcourt
Port Harcourt, Nigeria.

Emails
promise.enyindah@uniport.edu.ng
linda.oghenekaro@uniport.edu.ng
chigoziri.marcus@uniport.edu.ng

Proceedings Citation Format

Enyindah, P., Oghenekaro, L.U. & Marcus, C. B. (2021): A Gitbash-Generative Model For An Enhanced Query Processing In A Distributed Database System. Proceedings of the Accra Bespoke Multidisciplinary Innovations Conference. University of Ghana/Academic City University College, Accra, Ghana. December, 2021. Pp 11-24
www.isteams.net/ghanabespoke2021. DOI - https://doi.org/ 10.22624/AIMS/ABMIC2021P2

A Gitbash-Generative Model for an Enhanced Query Processing In a Distributed Database System

Enyindah, Promise, Oghenekaro, Linda Uchenna & Marcus, Chigoziri Bobby.

ABSTRACT

This research study is directly focused on an enhanced query processing system using GitBash-Deep Generative Model, it is use to improve the performance of big data. Due to Poor query execution plan and absence of an improved aggregate query method which impose threats to qualities of good query processing in distributed databases which also cause memory pressure in a database, inflated central processing unit (CPU) and an overall reduction in concurrency. Also the volume of data use in any organization is progressively increasing in it seize, thereby resulting or demand a large storage location or space system. The availability of intended software that will confront the problem of storage contents and execution plan for query processing is a vital challenge facing big data in a processing system, which can prompt to information loss and missing record of vital document in an enterprises. The study will boost solution and addresses the problem of latencies, inability to transform and store queries, and poor query processing plans for future improvement of distributed databases, using an unsupervised learning approach such as deep generative model. The adopted methodology is dynamic system development method (DSDM). php programming language and mongoDB database is adopted for implementation. The datasets used for the proposed system was generated from e-library server/data.json (query) and is inserted into MongoDB database json format. The datasets are all trained and recognized by the Deep Generative algorithm through the Git Bash server. The proposed system result shows highly increase in the efficiency of management of big data and improved query processing.

General Term
Query Processing System

Keywords
Big Data, Deep Generative Model, Query Processing System, Information Management, Gitbash.

1. INTRODUCTION

Query processing is an efficient activity that aids Database end-users to retrieve specific information from the database. Secondly, Query processing in a distributed system requires the transmission of data between computers in a network. The arrangement of data transmissions and local data processing is known as a distribution strategy for a query. In most parts of Nigeria, data is conventionally obtained by manual entries though some technical data is obtained using specialized metering but this form of entry is rarely automated. The study focused on the application of an improved Deep Generative Model for improving query processing in information management systems. A good query plan is needed for an improved performance of big data.

The conversion of useful data into information is a task mostly done by trained experts. Information is only useful if it can be converted into diverse beneficial forms and in a timely manner. Speech synthesis from text, audio-visual systemic converters, and sensor-aware. acquisitioned systems can result in very robust information systems. Real-time data in industrial environments is typically streaming and unstructured. The data is typically obtained sequentially over time. To obtain the relevant information is a key part of any modern information management system as time critical industrial systems need to be well informed to minimize losses or cost of operations, improve the working conditions and also create the enterprise. Information can be re-generated as new and passed via industrial control systems over long distances in-order to operate them more efficiently. Because information can be converted into different forms, it is well suited to generative models. Generative models can help build more efficient systems that are robust to making decisions without the usual cost implications in memory or any hand-engineered approach.

[1] presented an approach to expand real-time database query solutions further by using dynamic probabilistic models. Whether this approach is sufficient in itself remains to be tried and tested effectively. The ultimate aim of any information system is to obtain relevant messages or codes from noisy or contaminated data distributions. The origins of information theory could be traced to the works of Shannon [2] and has deep probabilistic roots. "Query" is a unique Database terminology that is used in Database Management Systems (DBMS). It is also a Database object. Simple algorithms are presented that derive distribution strategies which have minimal response time and minimal total time, for a special class of queries.

These optimal algorithms are used as a basis to develop a general query processing algorithm. Distributed query examples are presented and the complexity of the general algorithm is analyzed. The integration of a query processing subsystem into a distributed database management system is also discussed in the study. Deep-Generative models is inspired by two core machine learning disciplines – Genetic Algorithms and Generative Models. While Genetic Algorithms is a biological motivated model based on human genetics and evolution, generative models are basically statistically driven models used to probabilistically define a data generating process which may be stochastic or not [3].

The remainder of this paper discusses; 2. Brief overview of related works. 3. The methodologies. 4. GitBash-Deep Generative Model 5. Results and Discussion. 6. Conclusion. 7. Further work

2. RELATED WORKS

A Deep Generative Model is a powerful way of learning any kind of data distribution using unsupervised learning and it has achieved tremendous success in just few years. All types of generative models aim at learning the true data distribution of the training set so as to generate new data points with some variations. But it is not always possible to learn the exact distribution of data either implicitly or explicitly and so there is need to model a distribution which is as similar as possible to the true data distribution. However, neural networks can be used to model a function which can approximate the model distribution to the true distribution. In statistical classification, including machine learning, two main approaches are called the generative approach and the discriminative approach. These compute classifiers by different approaches, differing in the degree of statistical modeling. Terminology is inconsistent, but three major types can be distinguished, following[4].

The distinction between these last two classes is not consistently made; Jebara (2004) refers to these three classes as generative learning, conditional learning, and discriminative learning, but Ng and Jordan[5] only distinguishes two classes, calling them generative classifiers (joint distribution) and discriminative classifiers (conditional distribution or no distribution), not distinguishing between the latter two classes. Applying data mining in information gathering of big data in the educational sector will go a long way in positively effecting management and decision-making[6] Another cardinal point of this study is the concept of big data. A good information gathering algorithm will maximize computation power and algorithmic accuracy to gather, analyze, link and compare large datasets, to also enable the drawing of large datasets to identify patterns in order to make economic, social, technical and legal claims[7].

3. METHODOLOGIES

In order to achieve the set goal for this research work, the following methodologies were adopted

3.1 Experimental Research

Experimental research method is the straightforward experiment, involving the standard practice of manipulating quantitative, independent variables to generate statistically analyzable data. Generally, the system of scientific measurements is interval or ratio based. When we talk about 'scientific research methods', this is what most people immediately think of, because it passes all of the definitions of 'true science'. The researcher is accepting or refuting the null hypothesis. The results generated are analyzable and are used to test hypotheses, with statistics giving a clear and unambiguous picture [8]. This enables researchers to compare the two groups and determine the impact of the intervention following processes were considered: survey, questionnaires, and interview

3.2 Agile Method

Agile software design methodology is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations [9]. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like

I. Planning
II. Requirements Analysis
III. Design
IV. Coding
V. Unit Testing and
VI. Acceptance Testing.

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

4. GIT-BASH GENERATIVE MODEL QUERY PROCESSING SYSTEM'SARCHITECTURE

The proposed adopts an unsupervised learning approach that uses a Git-bash deep generative model shown in figure 1. The proposed system uses mongo DB for data storage, which is capable of distributing big data with the help of deep generative algorithm which assist in fast processing and searching processed of an unstructured data.

The data to be search are keyed into the search term. The sorted data will immediately display on the data structure. In this work, the system is presented with unlabeled, uncategorized data and the system's algorithms act on the data without prior training. The output is dependent upon the coded algorithms. The specification or Requirement for performing simulations on the proposed system is given in Table 1. The requirement specifications typically includes the components necessary for implementing a given software process. It also includes some key information about the type and nature of application domain (such as the use of generative domains e.g. the use of deep generative algorithm domain). It must be emphasized that these requirements may change depending on the Application domain.

The initial requirements specifications will hence include:

1. A formal definition of the primary components required for implementing the system (e.g. the use of genetic algorithm).
2. A description of the data attribute structure of the functional objects/or attributes in the software system (state some key data structures here) See below and use as appropriate.

Table 1 show the requirement specification data used for training and testing of the proposed system.

The proposed system uses Git-Bash Deep generative model for query processing system (artificial system) to evolve a set of system parameters. The proposed Systems components and dataflow diagram are as shown in figure 1 and 2; these includes:

Dataset

The dataset use is called data.json from the existing system e-library server/data.json (query). It is the crisp set for query input. The dataset contains all the unstructured big data from the cloud e-library server/data.json (query). The dataset was generated from existing system e-library server/data.json (query) and is inserted into the mongodb and send to gitbash server where command is been given and then sent to the gitbash- generative mode for conversion from its original crisp set to query set that is now understand by the user. The proposed model contained one thousand (1000) datasets extracted from more than one million dataset in the existing system e-library server/data.json (query).

Information or Files

This component holds the unstructured data that are inputted into the mongodb. These data are store in the mongodb interface before sending to the gitbash server. This component is also another interface where information is sending to the model. It holds information or file from different users for conversion from its original state to query set for proper understanding.

Mongo DB

This component receives all the dataset and the information of the dataset from their respective source and stores them and sends them to the git bash server. The mongodb is a large storage location use to store the information or files and the dataset from the existing system e-library server/data.json. The mongodb encrypts and secure the dataset. The operation perform by the mondodb is done in the MongoDB audit log's which displays the dataset on an html application. There are created tool in the MongoDB audit log's which allows the user to conduct multilevel search queries on the audit log data. The dataset goes into the MongoDB audit log's for auditing. The auditing system writes every audit event on the dataset to an in-memory buffer of audit events. MongoDB writes this buffer to disk periodically.

The auditing system writes and display these datasets to the log file in a JSON format. The json format of an audit dataset event is done by typing the attribute types (ex. string/int/timestamp), ("System Event Audit Messages"). This command then sends to the discriminative algorithm to apply rules and parses an array of these objects as its main data source. To execute complex nested queries, the mongodb uses a complex Boolean expression in form of a decision tree. For the nested query operation, the dataset has expressions and groups which denote a level of nesting. The query operation is done in the following format.

```
{
conditions:[A],
groups:[
{
conditions:[B, C],
groupOperator: 'AND',
groups:[{
conditions:[D,E]
groupOperator:OR
}]
}
]
}
```

In the JSON dataset format expressed on a tree as shown in code above, the root of the tree would be the first group operator with the condition A as the left leaf and an 'OR' node on the right for nested group's operator. This node is connected with the conditions inside of its group which is B ∧ C and D ∧ E. Having the query expressed like this makes it easy to walk the tree and convert the expression into Postfix format before processing the query. From the search query in a tree format above, the full query is: A and (B and C or (D and E)) Postfix query is: ABC AND DE AND OR AND.

The Git Bash servers

The gitbash server component is use to connects the big data from the Mongo DB to the discriminative algorithm. The gitbash server contains the query mechanism that is use to enables the dataset to accommodate the Microsoft window storage location which provide emulation layers. The gitbash server component is use to control and manage the big dataset in the window storage location and allows the users to issues different command and format to the big data. The operation of the git bash server is done by first run or lunch the git bash. The first lunching of the Git Bash, allows the Generative algorithm to have a link on the MongoDB platform, this is done by typing in the code: cd with a space type documents/query, then press enter and type npm with a space type run with a space type serve then press enter.

These commands will immediately link the Generative algorithm to have a link on the MongoDB platform, which is the first command execution in the improved query processing. The second operation on the git bash is by lunching it the second time, the second lunching of the git bash server is to connect the datasets in the MongoDB to the discriminative algorithm to display on the design interface. The second operation of the Git Bash server is done by simply right click on its icon and type in the code: cd with a spaces type documents/query/server then press enter and type node with a space type index.js then press enter. This code will immediately connect the algorithm and the MongoDB.

When the two operation of the git bash are running simultaneously at the same time, it will display data connection successful. This means that the connect to the program for execution successful.

Gitbash-Generative Model

This component receives the data set from the cloud e-library server/data.json and send to the generative adversarial network. The dataset then goes into the inference engine to assign the right rule that will be used to convert the scrip dataset by the algorithm. The gitbash generative model component contains some other interface which are inference engine, discriminative algorithm that perform the conversion of the scrip set to the query set.

The Generative Adversarial Network

This is a component in the gitbash-generative model that contains the inference engine and the discriminative algorithm. This component performs the main function of the conversion. It uses the unsupervised learning method on the data set. The rules are store in the inference engine.

The inference Engine

The inference engine component contains several rules use for conversion from the original scrip dataset to the require query set for querying the big data. The inference system retrieved rules from the rule base which then produce the require output query. Each of the rule determined the type of query needed to perform. Once the unstructured database is converted, the corresponding input query sets are passed to the inference engine that process current inputs using the rules retrieved from the rule base, then produces an outputs query set. This component contains rules use to train the algorithm depending on the types of data set.

The inference engine was used to build the model. The inference engine specifies the features of inputted datasets based on a given label in the application, which use the output probabilities from the Generative Adversarial Network to make decisions on the most likely variables or parameters that influence the data generating stage. It stores the value in a local variable, and then using that variable in the control condition. Local variable was used to hold the length of the logData. The rule use by the inference engine local variable control condition, which is;

```
for (var x=0, arrLength=logData.length; x<arrLength;x++){
//logic
 }
```

The Discriminative Algorithm

This component contains the algorithm use in the proposed model. When the data set in the inference engine assign the right rule to be used for training the algorithm, the algorithm will then act upon the dataset to produce the desire result. The discriminative algorithm evaluates the dataset by apply step that guide the conversion. When the json format of an audit dataset event, that is attribute types (ex. string/int/timestamp), (“System Event Audit Messages”) or command is sends to the discriminative algorithm, it then applies rules and parses an array of these objects as its main data source. The postfix array is then evaluated in a stack method to filter down the audit log data. The algorithm looped through the query one time and scanned through the log file for each condition to find matches.

Homogeneous Distributed Database

This component enables the converted query set to match with the corresponding answer of the request (i.e. prestored datasets in the database). Furthermore, the homogeneous distributed database system is a network of two or more databases (with same type of DBMS software) which can be stored on one or more machines on a network (nodes). So, in this system data can be accessed and modified simultaneously on several databases in the network.

Query Output Aggregation

This component derives group and subgroup query results by analysis of a set of individual data entries.

Query Results Output

This component displays all the queries in the mongo database storage to the user. It shows all the aggregate components of all the dataset at the same time.

Table 1: Sample Input/output Specifications for the Deep generative algorithm for Query Processing System

Field name	Data Type	Field Size/Width	Decimal	Index
Natrum carbonicum	Character	20	no	id 1
Rheum officinals	Character	15	no	id 2
Benzocaine	Character	20	no	id3
Sodium	Character	15	no	id4
Menthol	Character	15	no	id5
White Alder	Character	20	no	id6
Flouride	Character	20	no	id7
Ethanol	Character	20	no	id8
Pollen	Character	15	no	id9

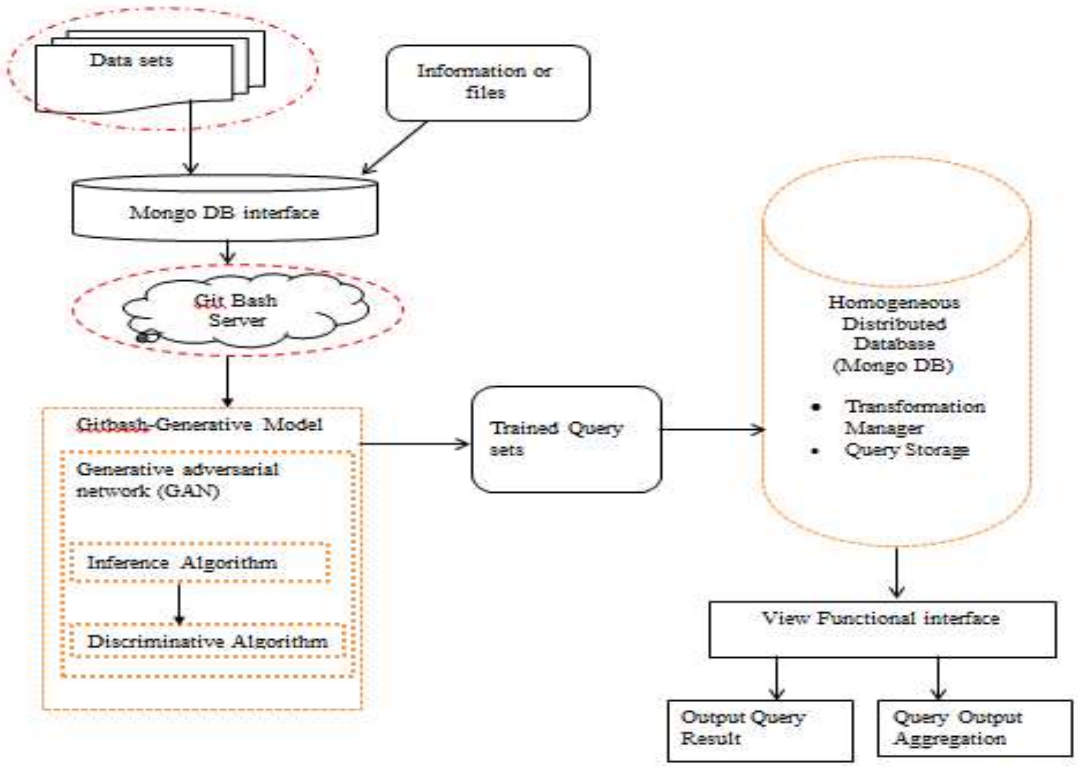

Figure 1: Architecture of the Proposed System

Figure 3 show the lunching of the Git Bash environment. This is the first command execution in the improved query processing. The lunching allows the Git-Bash Deep Generative algorithm to have a link on the MongoDB platform. In the lunching, type in the code: cd with a space type documents/query, then press enter and type npm with a space type run with a space type serve then press enter. These commands will immediately link the Deep Generative algorithm to have a link on the MongoDB platform. The function of lunching is to connect the datasets in the MongoDB to the Git-Bash Deep Generative algorithm to display on the design interface. To perform function on the Git Bash server simply right click on its icon and type in the code: cd with a spaces type documents/query/server then press enter and type node with a space type index.js then press enter. This code will immediately connect the algorithm and the MongoDB. After running the firs and the second Git Bash server click on start button and type in run and click on run at the top left side, this will immediately pop up another interface, then click on any item and press m, this will locate MongoDB install program, then right click and click on start. This will start running the MongoDB.

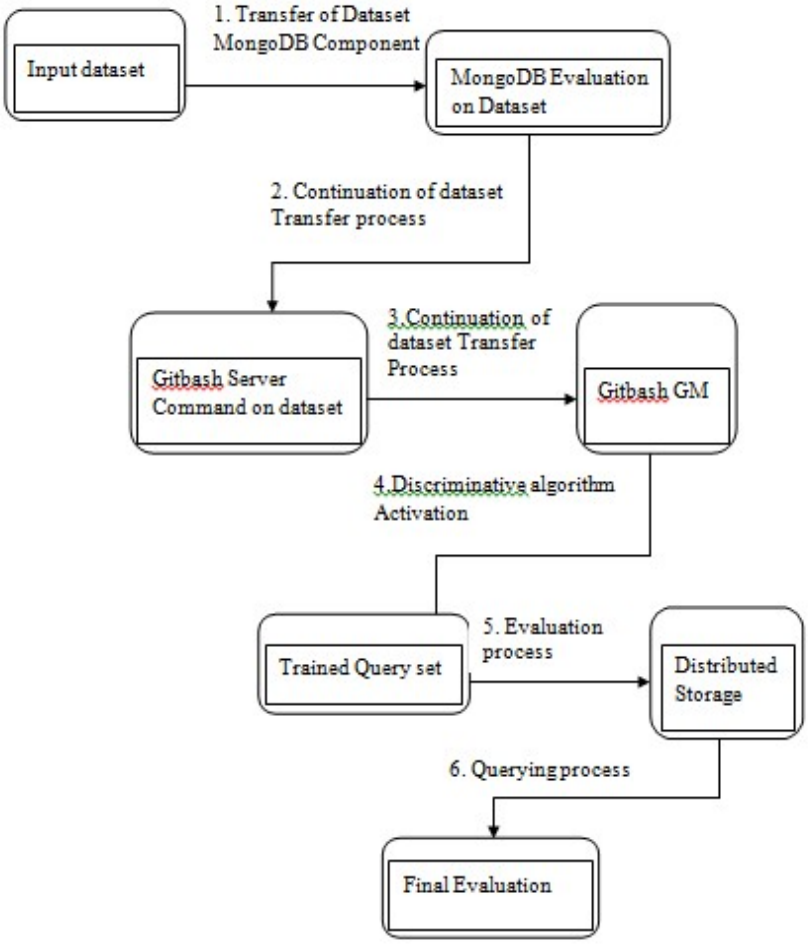

Figure 2: Dataflow Diagram for the Proposed System

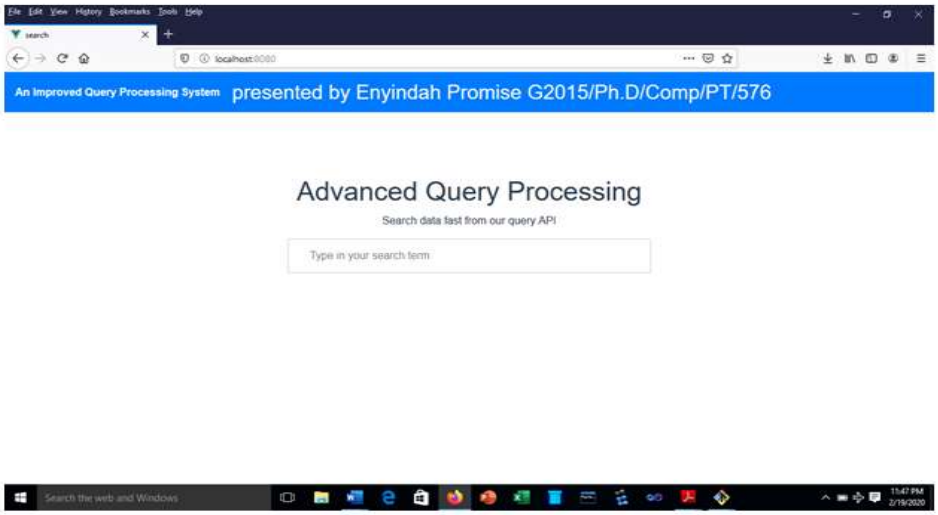

Figure 3: MongoDB not Run on Query

5. RESULT AND DISCUSSION

Table 2 and 3 show the performance ranking and evaluation of the git-bash generative model result for the proposed system. The variables used during coding are deep n, git-bash, search item, APL, journal, and mongodb. The values in the table were taken during runtime and are measure in second, the highest values recorded in the table 2 is 11. The graph of time against performance is plotted in figure 4 and figure 5 shows Data connection on Git Bash Server to Gitbash Generative Model

The result from the graph shows high increase rate of the variables in the proposed system when compared with the existing system. The parameters use in the graph includes processing speed, scalability, graphical user interface, availability and usability, query storage, and transformation ability. The graph is plotted efficiency against parameters. The highest value of the efficiency rate is 100. On the vertical axis (efficiency rate %) 20 units represent 1cm. the result from the graph shows the increase of each of the parameters in the proposed system which indicate an increase in performance, these shows that the performance ranking of the proposed system is of increase with better performance

Table 2: Performance Ranking of Query Results for the Proposed System

Deep G. Rank	Git Bash Rank	Search Item Rank	API Rank	Search Journal Rank	MongoDB Rank
8	4	2	3	12	9
10	3	3	11	4	5
11	6	9	6	7	6
9	10	4	5	8	3
10	9	10	8	9	10
9	5	7	9	6	8
Second (s) 05.7	7.06	05.09	04.09	03.04	03.04

Table 3: Performance Evaluation of the Proposed System

S/N	PARAMETERS	Efficiency Rate (%)
1	Processing Speed (PS)	98
2	Scalability (S)	92
3	Graphical User Interface (GUI) Quality	95
4	Availability and Usability (AU)	87
5	Query Storage and Transformation Ability	100

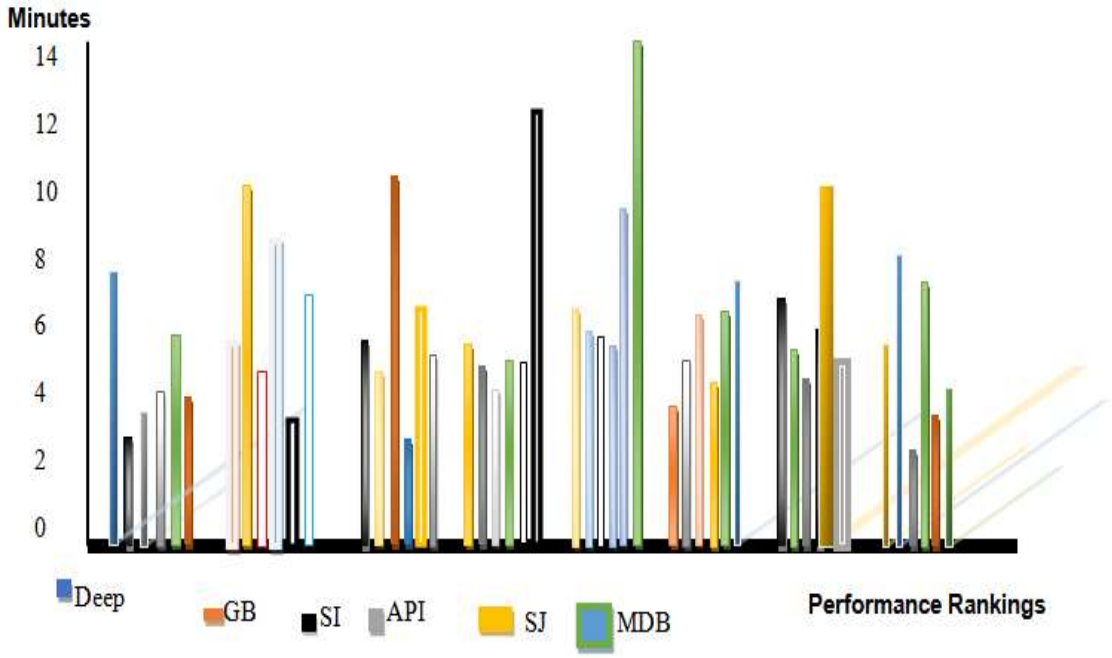

Figure 4: Query Results Performance Ranking Chart for the Proposed System

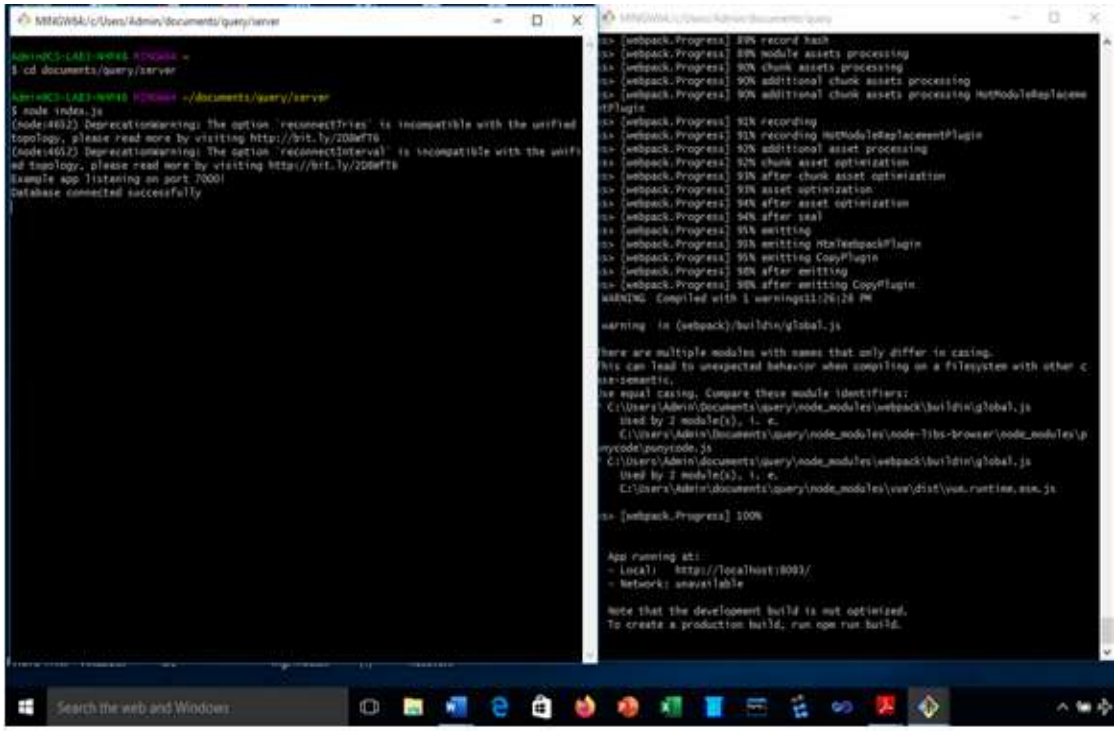

Figure 3: Data connection on Git Bash Server to Gitbash Generative Model

After series of runs and test using proposed Git-Bash Generative model system, test result results shown great improvement

i. Latency Reduction: This is because the proposed system uses a Git-bash deep generative model which consist of a hybridized algorithm (i.e. generative and discriminative) to arrive at a query conclusion that is reached on the basis of evidence and reasoning.
ii. Best Query Result for Structured and Unstructured Datasets: Git-Bash Deep Generative Models algorithms can be trained using different data formats, and still derive insights that are relevant to the purpose of its training. For this implies that the proposed Git-Bash deep generative models' algorithm can uncover any existing relations between pictures, social media chatter, industry analysis, weather forecast and more to predict future stock prices of a given company.
iii. No need for manual labeling of datasets before query processing:
iv. The proposed system supports self-automated query processing which also boycott the need for manual labeling of data. This is because; labeling process is simple but time-consuming. For example, labeling photos "dog" or "muffin" is an easy task, but an algorithm needs thousands of pictures to tell the difference. Other times, data labeling may require the judgments of highly skilled industry experts, and that is why, for some industries, getting high-quality training data can be very expensive.
v. An improved Graphical User Interface, Technique for Query storage and transformation for users of Homogeneous Distributed Database: The proposed system has enabled user-friendliness in the usage of homogeneous distributed database. In addition, users of the proposed system can be able to document, store and transform query sets in the distributed database

6. CONCLUSION

In this study, the researchers have presented an improved approach to query processing through the application of a Git-Bash deep generative model and mongodb. The improved approach depicts an unsupervised learning technique for query processing which is also a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data.
The findings of this study are recommended to database administrators and analysts in e-library environments, software developers and researchers with keen interest in the study area. This is because data management and request via queries are becoming complex day by day. In other words, the need for an improved query processing using Git-Bash Generative Model is highly indispensable.

7. FURTHER WORK

The limitations of the research should be improved in the study especially in a sophisticated application software device that will recognize real-time unstructured query data for homogeneous distributed databases. In addition, also improvement should integrate on other complex NoSQL databases such as Apache Cassandra, Hadoop and Mapreduce to the proposed system in future.

REFERENCES

[1] Shammana, J. (2011). A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization, In J. D. Schaffer, (ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, 51-60.

[2] Bengio, B. (2015), Couprie, M. & Valduriez, P. 2015. Overview of Parallel Architectures for
Database. *The Computer Journal*, 36, 734-740.

[3] Bennett O. (2018), Hybrid Technique for Optimization of Query Processing in a Distributed Database, *International Journal of Computer Science and Mathematical Theory (IJCSMT), 4(2), 19 – 27*

[4] Jebara A. (2004) Classifiers computed without using a probability model are also referred to loosely as discriminative. The distinction between these last two classes is not consistently
made; *International Journal of Scientific and Technology Research (IJSTR), 6(15), 29-56*

[5] Jordan N. (2004) generative classifiers (joint distribution) and discriminative classifiers (conditional distribution or no distribution), *International Journal of Scientific and Technology Research (IJSTR), 6(15), 42-66*

[6] Shafiq A. (2014), Data Mining Algorithms and their applications in Education Data Mining,
International Journal of Advanced Research in Computer Science and Management Studies (IJARCSMS) 2(7), 50 – 56

[7] Kelvin T. (2016), Big Data: Understanding Big Data, Engineering and Applied Science, *Aston University, England, Research Gate Publications, https://www.researchgate.net/publication/291229189, 56 – 61*

[8] Oppenheim, A. 1992. Questionnaire Design, Interviewing and Attitude Measurement, London, Pinter. Pp 303

[9] Gaurav, K. and Pradeep, K. 2012. Impact of Agile Methodology on Software Development Process: International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 2, Issue 4