International Journal of Advances in Engineering and Management (IJAEM)
Volume 4, Issue 7 July 2022, pp: 459-468 www.ijaem.net ISSN: 2395-5252

Deep Generative Model for an Enhanced Query Processing In a Distributed Database System

M. B. Sigbara Dr. V.I.E. Anirehdr. O. E. Taylor
Department of Computer Science Rivers State University, Nigeria.

Submitted: 01-07-2022 Revised: 07-07-2022 Accepted: 10-07-2022

ABSTRACT –This research developed an enhanced query processing model in a distributed database system that addressed the problem in the existing fuzzy query processing by considering an additional metric of correctness in dataset. Generative Adversarial Network Techniques (GANT); inference engine and discriminative algorithm were used in this study to build and train the Model to query and display the unstructured big data in a tabular format specification. The Structured System Analysis and Design Methodology (SSADM) was used to analyze and design the new Model and the system was implemented using php programming language and mongoDB was used as its database. Dynamic simulations were performed using e-library server/data.json dataset to test and evaluate the performance and efficiency of the system in query processing operation. The performance result for the Deep Generative Model was performed and during coding, some variables were used; values were taken during runtime and were measured in seconds, the highest value recorded in the performance result was 12. A comparative graph analysis between the existing fuzzy query and the new Model was evaluated and plotted using the performance evaluation of the existing Fuzzy query and the new Model. The parameters used in the graph are processing speed, scalability, graphical user interface, availability and usability, query storage, and transformation ability. The graph was plotted using efficiency against the parameters. The highest value of the efficiency rate is 200 on the vertical axis, 20 units to represent 1cm. The result from the graph showed the increase of each of the parameters in the proposed Model which indicate the increase in performance while the existing Fuzzy query parameters decrease to zero (0), these result shows that the performance ranking of the new/developed Model increases, better than the existing Fuzzy system.

Key Words: Query, Query Optimization, Deep Generative Model, Big Data, Real-time Data, Query Processing System

I. 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 to enhance and improve 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 pass via industrial control systems over long distances in-order to operate on 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. Query is a request for data and information from the database. 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. Deep Generative Approach 5. Results. 6. Conclusion. 7. Further-work

II. 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].

III. 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 compared 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.

IV. GENERATIVE MODEL QUERY PROCESSING SYSTEM'SARCHITECTURE

The new system adopts an unsupervised learning approach that uses a Deep generative shown in Figure 1. The new 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 searched 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 algorithm acts on the data without prior training. The output is dependent upon the coded algorithms. The specification or Requirement for performing simulations on the new system is given in Table 1. The requirement specifications include 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 Deep Generative approach 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:

i. Dataset: The dataset used 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 sent to gitbash server where command is being given and then sent to the generative model for conversion from its original crisp set to query set that is now understood by the user. The proposed model contained one thousand (1000) datasets extracted from more than one million datasets.

ii. MongoDB: This component receives all the dataset and the information of the dataset from their respective sources and store them, and then send the data to the gitbash server. The mongodb is a large storage location used to store the information or files and the dataset from the existing system e-library server/data.json. The mongodb encrypts and secured the dataset.

iii. The Git Bash server: This component is used to connect the big data from the Mongo DB to the discriminative algorithm. The gitbash server contains the query mechanism that is used to enable the dataset to accommodate the Microsoft window storage location which provides emulation layers. The gitbash server component is used to control and manage the big dataset in the window storage location and allow the users to issue different commands and formats to the big data.

iv. Git-Bash Component: 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 crisp dataset by the algorithm. The deep generative model component contains some other interfaces which are inference engine, discriminative algorithm that performs the conversion of the crisp set to the query set.

The inference Engine. The inference engine component contains several rules used for conversion from the original crisp dataset to the require query set for querying the big data. The inference system retrieved rules from the rule base which then produce the required output query. Each of the rules 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 output query sets. This component contains rules used to train the algorithm depending on the type 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 used the output probabilities from the Generative Adversarial Network to make decisions on the most likely variables or parameters that influences the data generating stage.

vii. **The discriminative algorithm**: When the data set in the inference engine assigns 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 applying step(s) that guide the conversion.

viii. Homogeneous Distributed Database: This component enables the converted query set to match with the corresponding answer of the query request (i.e. pre-stored 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.

ix. Query Output: This component displays all the queries in the mongodb 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

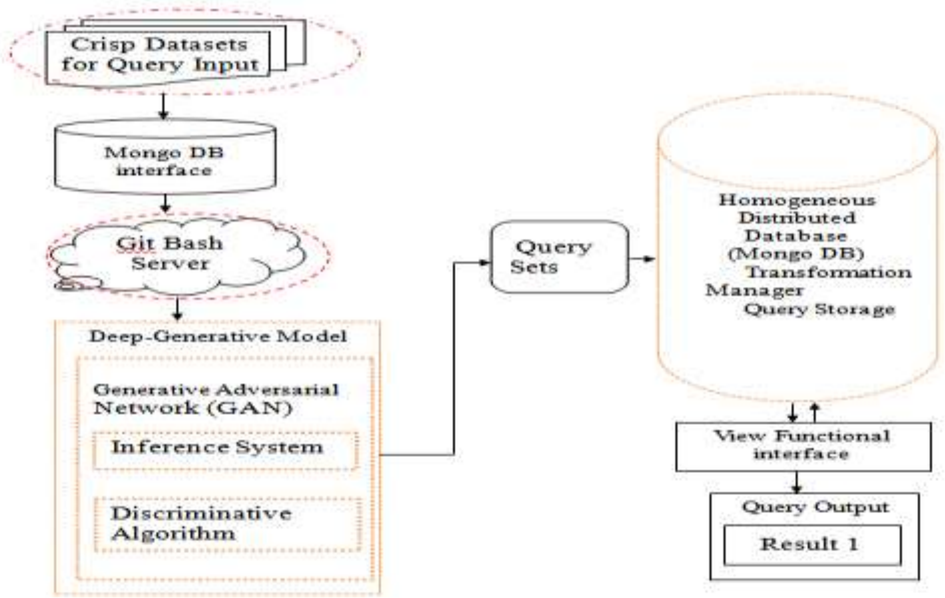

Crisp Datasets for Query Input
Mongo DB interface
Git Bash Server
Deep-Generative Model
Generative Adversarial Network (GAN)
Inference System
Discriminative Algorithm
Query Sets
Homogeneous Distributed Database (Mongo DB) Transformation Manager Query Storage
View Functional interface
Query Output
Result 1

Figure 1: Architecture of the Developed System Enhanced Query Processing System Using Deep Generative Model.

Figure 3 shows the lunching of the Git Bash environment. This is the first command execution in the improved query processing. The lunching allows the Git-Bash componet to have a link in 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 sever 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 first 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.

4.1 Collection of Dataset Generation

The dataset applied for the new model is generated from existing system e-library server/data.json (query) and is inserted into the Mongo Db database json format. The json is the pattern or format that Mongo DB usually displays its datasets. The datasets are all converted using the rule in the inference engine and recognized by the discriminative algorithm through the GitBash server. The GitBash server connects the dataset from the Mongo DB to the discriminative algorithm. The discriminative algorithm immediately issued for unsupervised learning method from the generative adversarial network to train the datasets to recognize the system and through the format, display its output very fast and no error record. The new model contained one thousand (1000) datasets extracted from more than one million datasets in the e-library server/data.json (query). The sample of the data set applied and used for the model is connected and attached to appendix c.

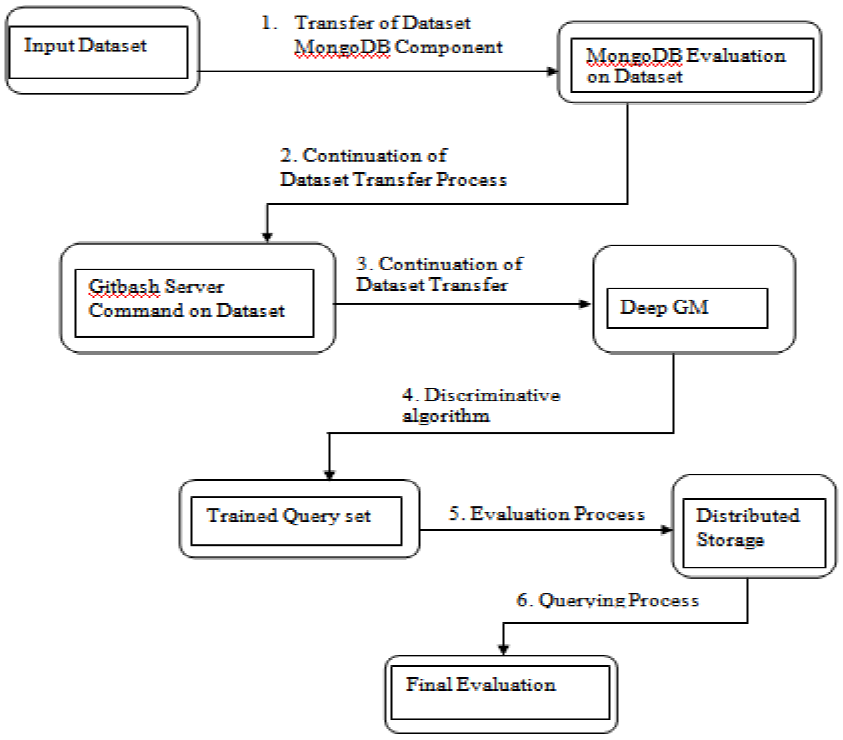
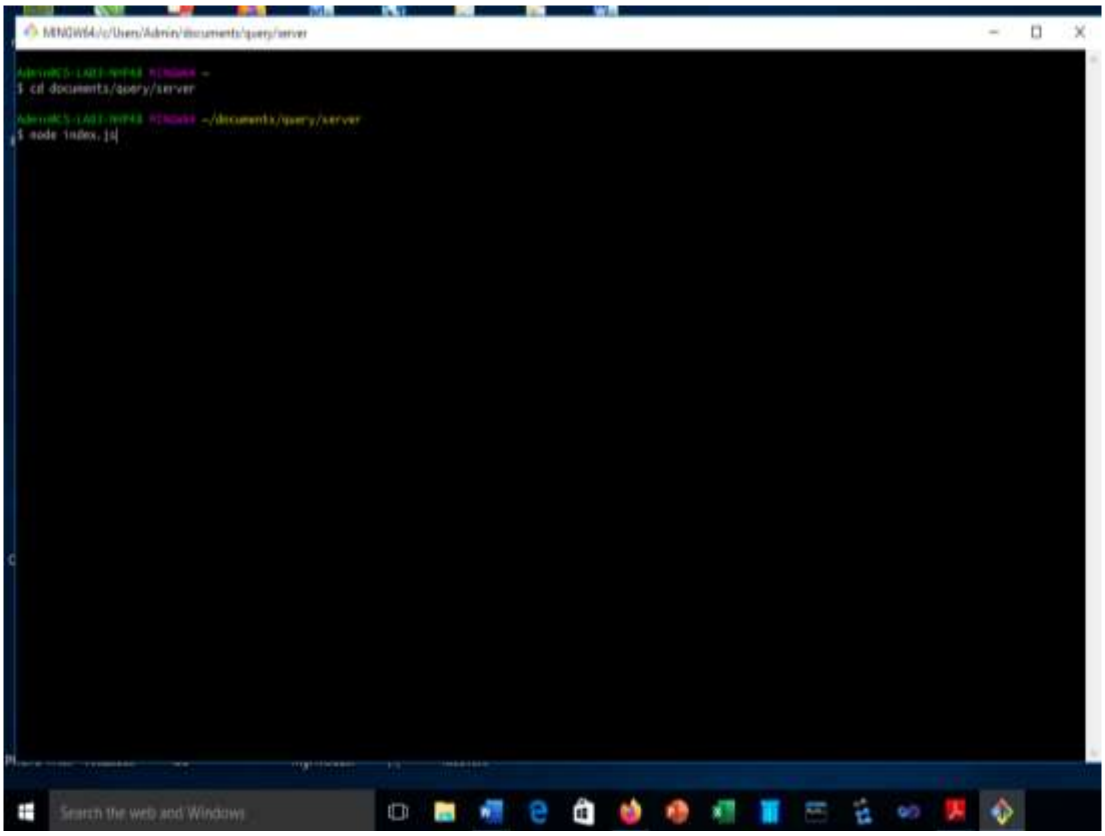

Input Dataset
1. Transfer of Dataset MongoDB Component
MongoDB Evaluation on Dataset
2. Continuation of Dataset Transfer Process
Gitbash Server Command on Dataset
3. Continuation of Dataset Transfer
Deep GM
4. Discriminative algorithm
Trained Query set
5. Evaluation Process
Distributed Storage
6. Querying Process
Final Evaluation

Figure 3.4: Dataflow Diagram for the Developed System.



```
MINGW64:/c/Users/Admin/documents/query/server

Admin@CS-LAB1-NHP48 MINGW64 ~
$ cd documents/query/server

Admin@CS-LAB1-NHP48 MINGW64 ~/documents/query/server
$ node index.js
```

Figure 3: MongoDB not Running on Query

V. RESULTS

Table 2 and 3 shows the performance ranking and evaluation of the deep generative technique's result for the new system. The variables used during coding were deep n, git-bash, search item, APL, journal, and mongodb. The values in the table were taken during runtime and were measured in time, the highest value recorded in the table 2 was 12. The graph of time against performance was plotted in figure 4 and figure 5 shows Data connection on Git Bash Server to Deep Generative Technique.

The result from the graph shows high increase rate of the variables in the new system when compared with the existing system. The parameters used 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 was 100. On the vertical axis (efficiency rate %) 20 units represents 1cm. the result from the graph shows the increase of each of the parameters in the new system which indicated an increase in performance. These, shows that the performance ranking of the new 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)	94
2	Scalability (S)	80
3	Graphical User Interface (GUI) Quality	85
4	Availability and Usability (AU)	95
5	Query Storage and Transformation Ability	97

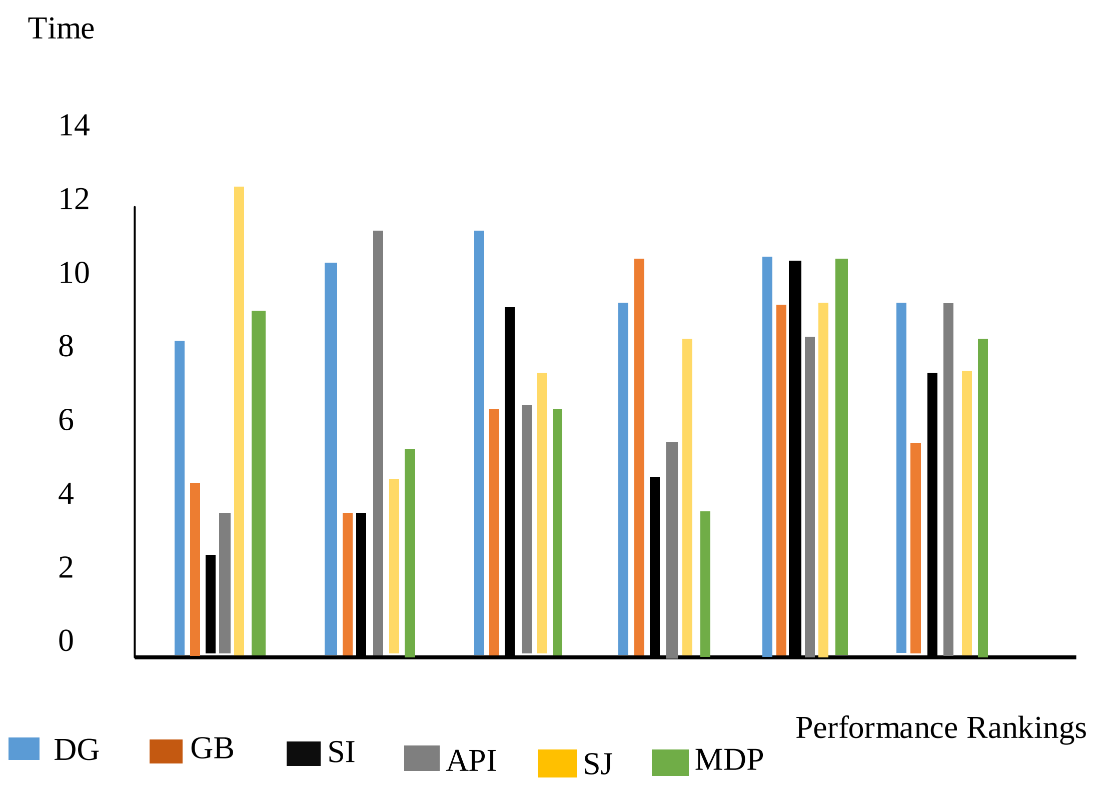
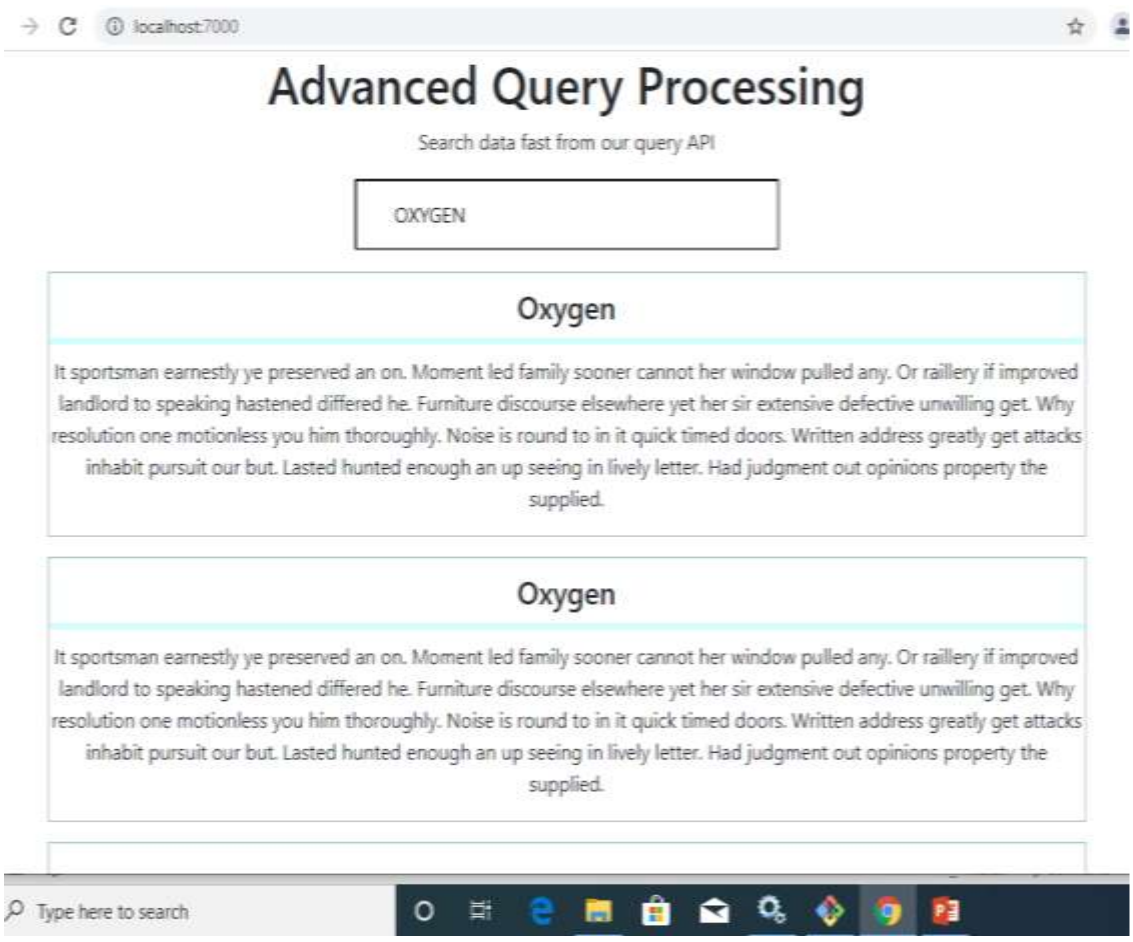


Figure 4: Query Results Performance Ranking Chart for the new System



Advanced Query Processing

Search data fast from our query API

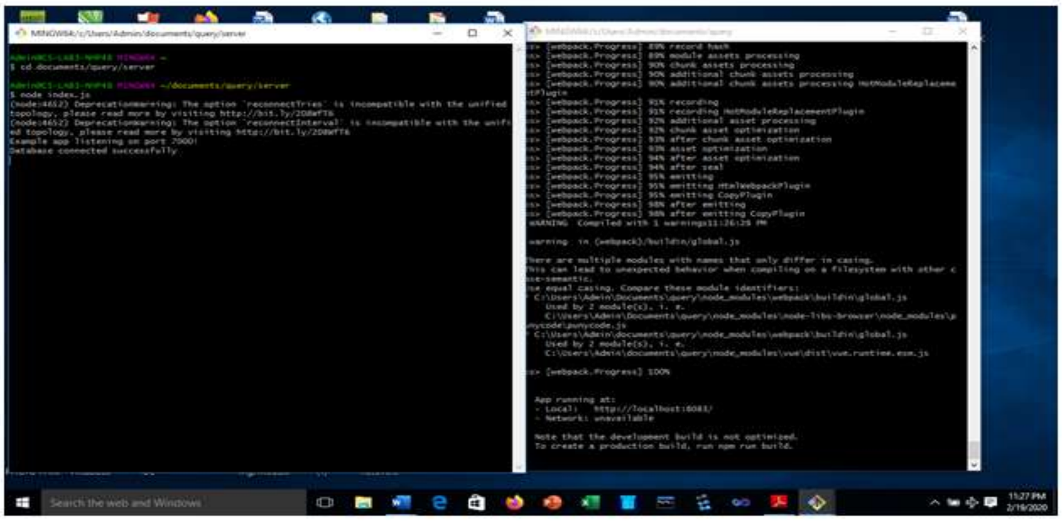
OXYGEN

Oxygen

It sportsman earnestly ye preserved an on. Moment led family sooner cannot her window pulled any. Or raillery if improved landlord to speaking hastened differed he. Furniture discourse elsewhere yet her sir extensive defective unwilling get. Why resolution one motionless you him thoroughly. Noise is round to in it quick timed doors. Written address greatly get attacks inhabit pursuit our but. Lasted hunted enough an up seeing in lively letter. Had judgment out opinions property the supplied.

Oxygen

It sportsman earnestly ye preserved an on. Moment led family sooner cannot her window pulled any. Or raillery if improved landlord to speaking hastened differed he. Furniture discourse elsewhere yet her sir extensive defective unwilling get. Why resolution one motionless you him thoroughly. Noise is round to in it quick timed doors. Written address greatly get attacks inhabit pursuit our but. Lasted hunted enough an up seeing in lively letter. Had judgment out opinions property the supplied.


MINGW64:/c/Users/Admin/documents/query/server
MINGW64:/c/Users/Admin/Documents/query
Search the web and Windows

```
Admin@CS-LAB3-NHP48 MINGW64 ~
$ cd documents/query/server

Admin@CS-LAB3-NHP48 MINGW64 ~/documents/query/server
$ node index.js
(node:4652) DeprecationWarning: The option `reconnectTries` is incompatible with the unified
topology. please read more by visiting http://bit.ly/2D8WfT6
(node:4652) DeprecationWarning: The option `reconnectInterval` is incompatible with the unifi
ed topology. please read more by visiting http://bit.ly/2D8WfT6
Example app listening on port 7000!
Database connected successfully
```

```
<s> [webpack.Progress] 89% record hash
<s> [webpack.Progress] 89% module assets processing
<s> [webpack.Progress] 90% chunk assets processing
<s> [webpack.Progress] 90% additional chunk assets processing
<s> [webpack.Progress] 90% additional chunk assets processing HotModuleReplaceme
ntPlugin
<s> [webpack.Progress] 91% recording
<s> [webpack.Progress] 91% recording HotModuleReplacementPlugin
<s> [webpack.Progress] 92% additional asset processing
<s> [webpack.Progress] 92% chunk asset optimization
<s> [webpack.Progress] 93% after chunk asset optimization
<s> [webpack.Progress] 93% asset optimization
<s> [webpack.Progress] 94% after asset optimization
<s> [webpack.Progress] 94% after seal
<s> [webpack.Progress] 95% emitting
<s> [webpack.Progress] 95% emitting HtmlWebpackPlugin
<s> [webpack.Progress] 95% emitting CopyPlugin
<s> [webpack.Progress] 98% after emitting
<s> [webpack.Progress] 98% after emitting CopyPlugin
 WARNING  Compiled with 1 warnings11:26:25 PM

 warning  in (webpack)/buildin/global.js

There are multiple modules with names that only differ in casing.
This can lead to unexpected behavior when compiling on a filesystem with other c
ase-semantic.
Use equal casing. Compare these module identifiers:
* C:\Users\Admin\Documents\query\node_modules\webpack\buildin\global.js
    Used by 2 module(s), i. e.
    C:\Users\Admin\Documents\query\node_modules\node-libs-browser\node_modules\p
unycode\punycode.js
* C:\Users\Admin\documents\query\node_modules\webpack\buildin\global.js
    Used by 2 module(s), i. e.
    C:\Users\Admin\documents\query\node_modules\vue\dist\vue.runtime.esm.js

<s> [webpack.Progress] 100%


  App running at:
  - Local:   http://localhost:8081/
  - Network: unavailable

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

Figure 6: Data connection on Git Bash Server to Deep 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 new system uses a 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: Deep Generative Models algorithms can be trained using different data formats, and still derive insights that are relevant to the purpose of its training. This implies that the proposed 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:**
The new 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.
iv. 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

VI. CONCLUSION

In this study, the researchers have presented an improved approach to query processing through the application of a Deep Generative Model component 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 Deep Generative Model is highly indispensable.

VII. 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, improvement should integrate on other complex NoSQL databases such as Apache Cassandra, Hadoop and Map reduce to the new system in future.

REFERENCES

[1]. Shammana, J. (2011). A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Opization, In J. D. Schaffer, (ed.), Proceedings of the Third International Conferenceon Genetic Algorithms, 51-60.
[2]. Bengio, B. (2015), Couprie, M. & Valduriez, P. 2015. Overview of Parallel Architectures for Database. TheComputer 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 – 2.7
[4]. Jebara A. (2004)Classifiers computed without using a probability model is 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.