ISSN(Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 5, May 2019

Correspondence Methods for SQL Query Clustering

Miss. Pooja Subhash Joshi[1], Prof. Dinesh D. Patil.[2]

PG Scholar, Department of Computer Science & Engineering, Hindi Seva Mandal's, Shri Sant Gadge Baba College of Engineering & Technology, Bhusawal, Dist. – Jalgaon, Maharashtra, India[1]

Associate Professor& Head, Department of Computer Science & Engineering, Hindi Seva Mandal's, Shri Sant Gadge Baba College of Engineering & Technology, Bhusawal, Dist. – Jalgaon, Maharashtra, India[2]

ABSTRACT: Database entrance logs are the initial point for various forms of database administration, from database performance tuning, to defense auditing, to standard design, and many more. Unfortunately, query logs are also bulky and clumsy, and it can be tricky for an analyst to extort broad patterns from the set of queries initiate therein. Clustering is a usual first step towards understanding the enormous query logs. However, many clustering methods rely on the concept of pair wise comparison, which is difficult to compute for SQL queries, particularly when the underlying data and database diagram is unavailable. The exploration of problem is computing comparison between queries, relying only on the query constitution. Accomplish an exact evaluation of three query similarity heuristics projected in the literature applied to query clustering on several query log datasets, expressive different kinds of query workloads. To develop the precision of the three heuristics, proposition a generic feature engineering approach, using standard query rewrites to normalize query structure. The proposed system approach is to generate important results in development for the performance of all similarities.

KEYWORDS: - SQL query, Clustering, Feature engineering, Query logs, Query workloads.

I. INTRODUCTION

Database systems (DBMSs) have been widely organized in latest years and their uses have happen to progressively more difficult and diverse. Physical design modification has consequently become more appropriate than ever before and currently database administrators expend substantial time tuning a sub-optimal installation for performance. As a result of this inclination, automatic physical tuning became a significant research crisis.

Database entrée logs are used in an extensive variety of settings, including calculating database performance alteration [2], yardstick development [3], database auditing [4], and compliance corroboration [5]. Also, many consumer centric schemes exploit query logs to help users by providing suggestion and personalizing the user understanding [6], [7], [8], [9], [10], [11]. As the basic unit of interface between a database and its consumers, the series of SQL queries that a user issues efficiently models the user's performance. Queries that are comparable in formation imply that they might be issued to execute similar duties. Examining a record of the queries examined by a database can facilitate database administrators with change, or assist safety analysts to evaluate the possibility and/or degree of a security breach. However, logs from endeavor database systems are far too huge to inspect physically.

The precise description of similarity may rely on, the pleased of the log, the database plan, database records, and frequent other information, some of which may not be accessible to directly when first starts examine the log. It is also expected that some of this information, similar to the exact contents of the database or even the database schema may not even be accessible for reasons of isolation or safety. As a result, this type of log study can speedily become a boring, lengthy process [11]. An earlier work of effort to address this difficulty for OLAP operations by performing query log analysis and examination. Within the scope of this, focus on study of SQL queries as a replacement for of OLAP queries. In exacting, lay the foundation for a more preset approach to SQL query log examination based on hierarchical clustering. Given a hierarchical clustering of the SQL query log, can physically alter how insistently the

ISSN(Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 5, May 2019

log is summarized. Choose a suitable level of granularity lacking a priori needing to identify accurately what comprise a comparable query.

The main focus of this paper is to modify the appropriateness of three presented query expanse metrics [13], [14], [15] to be used with hierarchical algorithms for clustering query logs. All of these metrics function on the query formation and do not depend on the convenience of underlying data or schema, thus assembly them relevant in an extensive variety of practical settings. Estimate the three metrics on two types of data: Human-authored and Machine-generated. Thus, using a suitable similarity metric, one can bunch the queries to acquire a significant clustering of the query log.

II. LITERATURE SURVEY

1) **Gokhan Kul, Graduate, Duc Thanh Anh Luong, Ting Xie, Varun Chandola, Oliver Kennedy, and Shambhu Upadhyaya (2018),** In this paper, database access logs are the starting point for many forms of database administration, from database performance tuning, to security auditing, to benchmark design, and many more Clustering is a natural first step towards understanding the massive query logs. However, several clustering techniques rely on the idea of pair wise similarity, which is difficult to compute for SQL queries, particularly when the fundamental data and database schema is unavailable. Authors investigate the problem of computing similarity between queries, relying only on the query structure. Author's way an exact view of three query similarity heuristics projected in the literature applied to query clustering on many query log datasets, instead of various types of query workloads. To improve the accuracy of the three heuristics, Authors propose a generic feature engineering strategy, using classical query rewrites to standardize query structure. The proposed tactic results in a significant enhancement in the performance of all three similarity heuristics.

2) **Kennedy, J. Ajay, G. Challen, and L. Ziarek (2015)**, In this paper authors focused on embedded database engines such as SQLite provide a convenient data persistence layer and have spread along with the applications using them to a lot of types of systems, as well as interactive devices such as smartphones. Android, the nearly all widely-distributed smartphone platform, both uses SQLite inside and provides interfaces cheering apps to use SQLite to store their own private structured data. As parallel functionality emerge in all main mobile operating systems, entrenched database performance affects the response times and resource expenditure of billions of smartphones and the millions of apps that sprint on them—making it more significant than ever to distinguish smartphone embedded database workloads. To do so, present results from an experiment which recorded SQLite activity on 11 Android smartphones during one month of typical usage. Our study shows that Android SQLite practice makes queries and entrée patterns quite different from canonical server workloads.

3) **G. Kul, D. Luong, T. Xie, P. Coonan, V. Chandola, O. Kennedy, and S. Upadhyaya (2016),** Insider threats to databases in the financial sector have become a very serious and pervasive security problem. This paper suggests a structure to analyze access patterns to databases by clustering SQL queries subjected to the database. Our project Ettu works by combination queries with additional similarly structured queries. The small number of intent groups that result can then be efficiently labeled by human operators. We illustrate how our system is intended and how the mechanisms of the system work. Our beginning results show that our system precisely models user intent.

4) **V. H. Makiyama, M. J. Raddick, and R. D. Santos (2015), SkyServer** the portal for the Sloan Digital Sky Survey (SDSS) catalog, provides data access tools for astronomers and scientific education. One of the boundaries allows users to go into ad hoc SQL statements to query the catalog, and has logged over 280 million queries since 2001. This paper explains text mining techniques and preliminary results on mining the logs of the SQL queries submitted to SkyServer, along with what other applications we foresee for such procedure.

ISSN(Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 5, May 2019

5) **R. Agrawal, R. Rantzau, and E. Terzi(2006)**, Aim to rank the tuples returned by the SQL query based on the context. They generate a rule set for contexts and assess the result of queries that fit in to the context according to the rule set. They imprison background and query as feature vectors and capture similarity through cosine distance between the vectors.

6) **G. Chatzopoulou, M. Eirinaki, S. Koshy, S. Mittal, N. Polyzotis, and J. S. V. Varman(2011)**, Aim to assist non-expert users of scientific databases by tracking their querying behavior and generating personalized query recommendations. They analyze an SQL query into a bag of fragments. Each different fragment is an attribute, with a weight allocated to it representing its importance. Each characteristic has two types of importance: (1) inside the query and (2) for the general workload. Similarity is defined upon common vector-based measures such as cosine similarity. A summarization/user profile for this approach is just a sum over all single query feature vectors that belong to their workload.

7) **X. Yang, C. M. Procopiuc, and D. Srivastava(2009)**, In this paper author says, on the other hand, build a graph following the query log by connecting associations of table attributes from the input and output of queries which are then used to compute the likelihood of an attribute appearing in a query with a similarity function like Jaccard coefficient. Their wish is again to help users in writing SQL queries by analyzing query logs.

8) **Giacometti, P. Marcel, E. Negre, and A. Soulet(2009)**, Similarly, aim to make recommendations on the discoveries made in the previous sessions for users to spend less time on investigating similar information. They introduce difference pairs in order to measure the relevance of the previous discoveries. Difference pairs are essentially the result column that is not included in the other return results; hence the method depends on having access to the data.

9) **K. Stefanidis, M. Drosou, and E. Pitoura(2009)**, Takes a different approach, and instead of recommending candidate queries, they recommend tuples that may be of interest to the user. By responsibility so, the users may make a decision to modify the selection criteria of their queries in order to include these results. Sapia [5] creates a model that learns query templates to prefetch data in OLAP systems based on the user's past activity.

10) **N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu(2010)**, On the other hand, is a context-aware SQL-autocomplete system that helps database users to write SQL queries by suggesting SQL snippets. In particular, it allocates a possibility score to every subtree of a query based on the subtree's regularity in a query log. These possibilities are used to find out the most likely subtree that a user is attempting to create, at interactive speeds.

11) **K. Aouiche, P.-E. Jouve, and J. Darmont(2006),** In this paper authors developed the first work we encountered that proposes a pairwise similarity metric between two SQL queries although it is not the aim of their work. They want to optimize outlook selection in warehouses by the queries pretense to the system. They think the assortment, joins and group by items in the query to generate vectors and use Hamming Distance to calculate how parallel two queries are. While produce the vector, it doesn't subject if an item emerge more than one time or wherever the item is. They cluster a similar query that creates a workload on the system and base their view creation strategy in the system on the clustering result.

12) **J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, and E. Turricchia(2014)**, Study various approaches to defining a similarity function to compare OLAP sessions. They center on comparing gathering similarity while also performing a review on query similarity metrics. They recognize assortment and link items as the most applicable components in a query pursued by the group by set. Inspired by the findings, they propose their own query similarity metric which considers projection, group-by, and selection-join items for queries issued on OLAP data cubes. OLAP data cubes are multidimensional models, and they have hierarchy levels

ISSN(Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 5, May 2019

for the same attributes. Compute the distance between the characteristics on unusual ladder levels, and compute the set comparison for projection, group-by, and selection-join sets independently when evaluating two queries. In our research, since we do not think the hierarchy levels in an OLAP system but center on databases, we consider all queries are on the similar level in the scheme to alter the formulas accessible in the paper. Namely, compute set similarity of projection, group-by, and selection-join sets of two queries with Jaccard coefficient. Also, Authors provide the flexibility to adjust weights of the three feature sets based on the domain needs.

III. PROPOSED SYSTEM

3.1. Feature Engineering

The grammar of SQL is declarative. By propose, users can write down queries in the technique they feel most relaxed, leasing entrenched correspondence rules dictate a final estimate approach. As a result, various syntactically different queries may still be semantically equivalent. However, we can still extensively develop clustering feature by normalize firm SQL features into a more usual form with methods such as official names and aliases, eradicate syntactic sugaring, and normalize nested query predicates. This method of regularization aspires to generate a new query that is more possible to be structurally related to other semantically similar queries.

3.2. Quality Metrics

We establish the superiority events and workloads to estimate three query correspondence methods and the feature engineering scheme. Our aim is to estimate how well a query correspondence method detains the assignment at the back a query with and without regularization. We use two types of real world query workloads: human- and machine-generated. We guess the difficulty of query similarity to be harder on human-generated workloads, as queries produced by machines are more possible to pursue a exacting, firm structural model.

3.3. KNN clustering

The purpose of this clustering technique is to just split the data based on the implicit similarities between different classes. Thus, the classes can be discriminate from one a different by penetrating for similarities between the data provides.

3.4. Proposed Framework

The proposed system is the connection between the information system and the user. It includes the increasing requirement and procedures for data research and those steps are essential to place deal data in to a utilizable form for processing can be achieved by inspecting the computer to understand data from a on paper or in print file or it can occur by having people keying the information directly into the system. The propose of input center of attention on controlling the quantity of input requisite, calculating the errors, avoiding stoppage, avoiding additional steps and custody the procedure easy The input is designed in such a way so that it offers safety and simplicity of use with maintain the confidentiality. Proposed method considered the following things:

- What information should be specified as enter?
- How the data should be approved or implied?
- The dialog to direct the working personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

Nothing of the similarity metrics implement as well as preferred, so propose and approximate a pre-processing step to produce more typical, identical query representations by leveraging query correspondence rules and data partitioning process. These rules are generally used by database management systems when parsing and assessing SQL queries**.** This method expansively recovers the attribute of all three distance metrics. Observe and identify sources of errors in the clustering process. Concretely, the explicit contributions of this article are:

- A estimate of presented SQL query similarity metrics
- An evaluation of these metrics on various query logs, and
- Applying query equivalence techniques to get improved query clustering accuracy.

ISSN(Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 5, May 2019

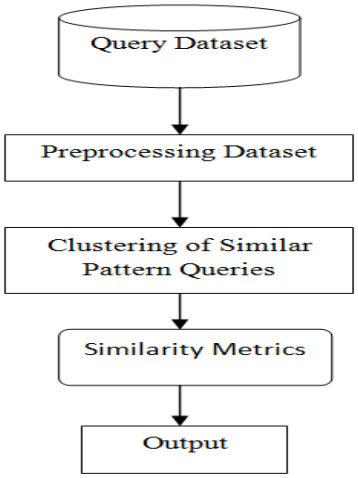


Fig. 1:- Proposed Framework

IV. EXPERIMENTAL RESULTS

A quality output is one, which meets the necessities of the user and presents the information evidently. In any system results of processing are correspond to the users and to other system all the way through outputs. In output design it is resolute how the information is to be move for direct need and also the hard copy output. It is the most essential and straight resource information to the user. Capable and intelligent output design recovers the system's relationship to facilitate user decision-making.

Dataset contain one or more Data Table, in this system we used three tables name as dataset, preprocess and result.

YY	MM	DD	HH	MI	SS	SEQ	THETIME	LOGID	CLIENTIP	REQUESTC	SERVER	DBNAME	ACC	ELAPSED	BUSY	NRROWS	STATEMEI	ERROR
2004	1	14	14	54	49	4114837	14-JAN-04	2003	212.68.72.	skyserver.	SDSSDR1	BESTDR1	public	0.03	0	1	select * fr	0
2004	1	14	14	55	26	4114836	14-JAN-04	2003	130.183.3.	skyserver.	SDSSDR1	BESTDR1	public	0.063	0.001	34	select * fr	0
2004	1	14	17	12	53	4114585	14-JAN-04	2003	218.1.188.	SkyQA	SDSSDR1	BESTDR1	public	0.016	0	16	select nan	0
2004	1	14	2	19	39	4117490	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.013	0	1	SELECT sp	0
2004	1	14	2	19	41	4117488	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0	1	SELECT sp	0
2004	1	14	2	19	42	4117486	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0	1	SELECT sp	0
2004	1	14	2	19	41	4117487	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.03	0.001	1	SELECT sp	0
2004	1	14	2	19	42	4117485	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0	1	SELECT sp	0
2004	1	14	2	19	44	4117483	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0	1	SELECT sp	0
2004	1	14	2	19	43	4117484	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0	1	SELECT sp	0
2004	1	14	2	19	45	4117481	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0	1	SELECT sp	0
2004	1	14	2	19	44	4117482	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.03	0	1	SELECT sp	0
2004	1	14	2	19	46	4117479	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0	1	SELECT sp	0
2004	1	14	2	19	45	4117480	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.033	0	1	SELECT sp	0
2004	1	14	2	19	47	4117478	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0	1	SELECT sp	0
2004	1	14	2	19	48	4117476	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0	1	SELECT sp	0
2004	1	14	2	19	47	4117477	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0	1	SELECT sp	0
2004	1	14	2	19	48	4117475	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0	1	SELECT sp	0
2004	1	14	2	19	50	4117473	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0	1	SELECT sp	0
2004	1	14	2	19	49	4117474	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0	1	SELECT sp	0
2004	1	14	2	23	8	4117133	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0	0	1	SELECT sp	0
2004	1	14	2	23	10	4117131	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0.001	1	SELECT sp	0
2004	1	14	2	23	11	4117129	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0.016	0	1	SELECT sp	0
2004	1	14	2	23	10	4117130	14-JAN-04	2003	131.215.1	skyserver.	SDSSDR1	BESTDR1	public	0	0	1	SELECT sp	0

Fig 1:- Dataset of proposed Framework

ISSN(Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 5, May 2019

First important task to start the system is to login as Admin by registered user name and password.

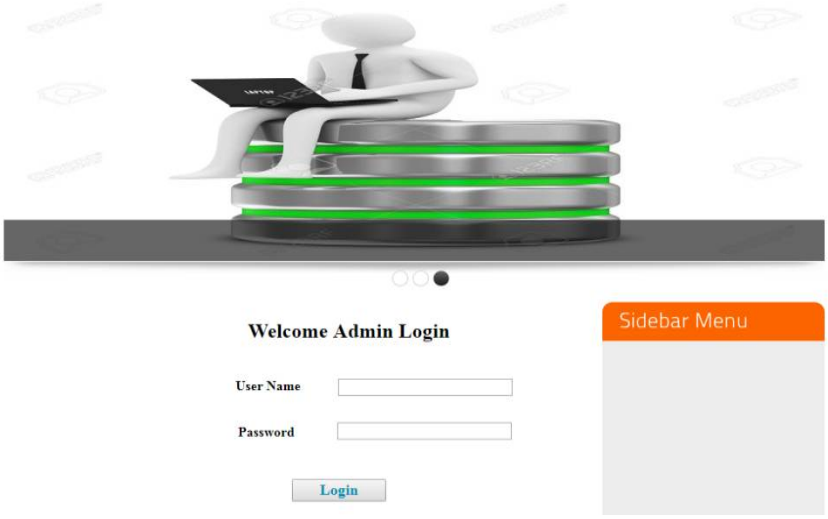


Fig. 2:- Admin Login

Admin has a menu bar which helped to perform various activities on dataset.

Fig 3:- Admin Menu

View the given dataset and know the data present in that table.

IJIRSET

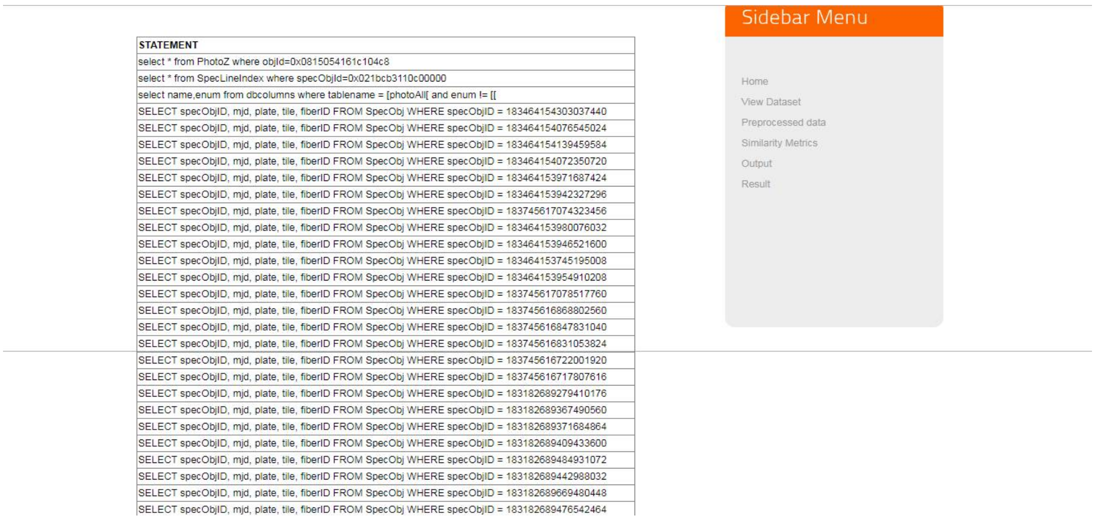
ISSN(Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

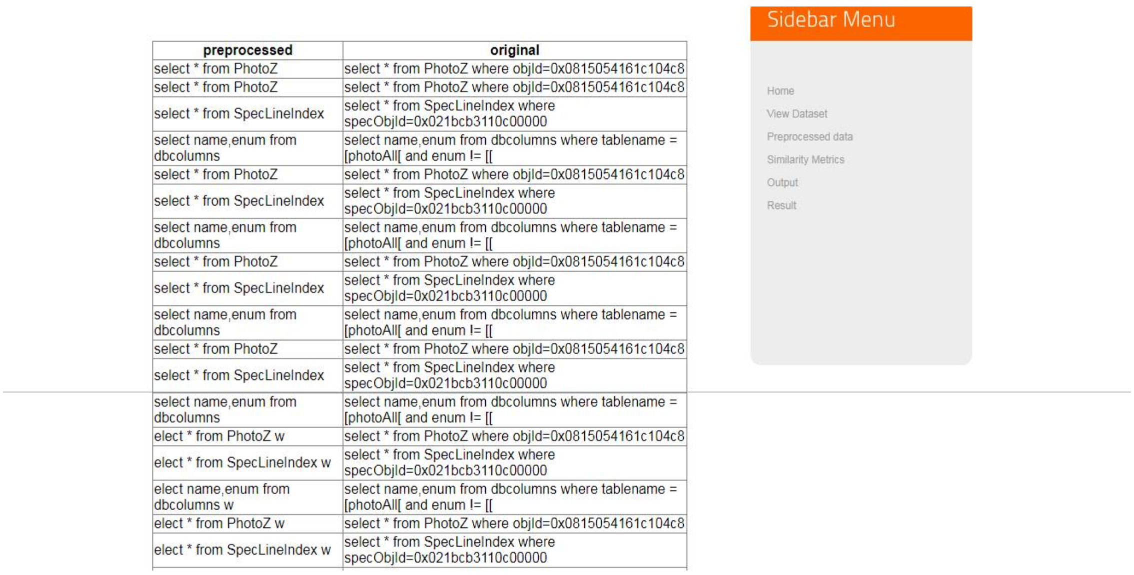
Vol. 8, Issue 5, May 2019



STATEMENT
select * from PhotoZ where objId=0x0815054161c104c8
select * from SpecLineIndex where specObjId=0x021bcb3110c00000
select name,enum from dbcolumns where tablename = [photoAll[and enum != [[
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183464154303037440
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183464154076545024
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183464154139459584
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183464154072350720
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183464153971687424
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183464153942327296
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183745617074323456
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183464153980076032
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183464153946521600
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183464153745195008
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183464153954910208
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183745617078517760
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183745616868802560
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183745616847831040
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183745616831053824
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183745616722001920
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183745616717807616
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183182689279410176
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183182689367490560
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183182689371684864
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183182689409433600
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183182689484931072
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183182689442988032
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183182689669480448
SELECT specObjID, mjd, plate, tile, fiberID FROM SpecObj WHERE specObjID = 183182689476542464

Fig 4:-View Dataset

Perform the Preprocessing on the given data and generate original query from dataset.



preprocessed	original
select * from PhotoZ	select * from PhotoZ where objId=0x0815054161c104c8
select * from PhotoZ	select * from PhotoZ where objId=0x0815054161c104c8
select * from SpecLineIndex	select * from SpecLineIndex where specObjId=0x021bcb3110c00000
select name,enum from dbcolumns	select name,enum from dbcolumns where tablename = [photoAll[and enum != [[
select * from PhotoZ	select * from PhotoZ where objId=0x0815054161c104c8
select * from SpecLineIndex	select * from SpecLineIndex where specObjId=0x021bcb3110c00000
select name,enum from dbcolumns	select name,enum from dbcolumns where tablename = [photoAll[and enum != [[
select * from PhotoZ	select * from PhotoZ where objId=0x0815054161c104c8
select * from SpecLineIndex	select * from SpecLineIndex where specObjId=0x021bcb3110c00000
select name,enum from dbcolumns	select name,enum from dbcolumns where tablename = [photoAll[and enum != [[
select * from PhotoZ	select * from PhotoZ where objId=0x0815054161c104c8
select * from SpecLineIndex	select * from SpecLineIndex where specObjId=0x021bcb3110c00000
select name,enum from dbcolumns	select name,enum from dbcolumns where tablename = [photoAll[and enum != [[
elect * from PhotoZ w	select * from PhotoZ where objId=0x0815054161c104c8
elect * from SpecLineIndex w	select * from SpecLineIndex where specObjId=0x021bcb3110c00000
elect name,enum from dbcolumns w	select name,enum from dbcolumns where tablename = [photoAll[and enum != [[
elect * from PhotoZ w	select * from PhotoZ where objId=0x0815054161c104c8
elect * from SpecLineIndex w	select * from SpecLineIndex where specObjId=0x021bcb3110c00000

Fig 5:- Preprocessing Data

Perform the similarity metrics on preprocessing data which gives the same queries occur after task performance.

ISSN(Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 5, May 2019

preprocessed
delete from company
delete from db
delete from details
delete from employee
elect * from PhotoZ w
elect * from PhotoZ where objl
elect * from SpecLineIndex w
elect * from SpecLineIndex where s
elect name,enum from dbcolumns w
elect name,enum from dbcolumns where tablen
insert into database(name,pass,email)
insert into details
insert into reg
insert into reg(name, pass, email)
select company
select db
select details
select * reg
select * from empid=656545
select * from employee
select * from field
select * from photoobjectance
select * from photoz
select * from register
select * from register table
select * from speclineindex
select * from student and employee
select fname

Fig 6:- Similarity Metrics

Output shows the clustering of similar pattern queries after performing similarity measures on dataset.

output
delete from company where id = hgdgfd675
delete from db where id = 658sfdgg
delete from details where id = fhshf675fg
delete from employee where id = fhshf675fg
elect * from PhotoZ w where objid = 0x0815054161c104c8,0x081505b681b40102,0x0815054161c104c8,0x0815054161c105c8,0x0815054161c105c9,0x0815054161c105c10,0x0815054161c105c11,0x0815054161c105c12,0x0
elect * from PhotoZ where objl where objid = 0x0815054161c104c8,0x081505b681b40102,0x0815054161c104c8,0x0815054161c105c8,0x0815054161c105c9,0x0815054161c105c10,0x0815054161c105c11,0x0815054161c10
elect * from SpecLineIndex w where specobjid = 0x021bcb3110c00000,0x021bcb3110c00011,0x021bcb3110c0012,0x021bcb3110c00001,0x021bcb3110c00002,0x021bcb3110c00003,0x021bcb3110c00004,0x021bcb3110c00005,0x021bcb3110c00006,0x021bcb3110c00007,0
elect * from SpecLineIndex where s where specobjid = 0x021bcb3110c00000,0x021bcb3110c00011,0x021bcb3110c0012,0x021bcb3110c00001,0x021bcb3110c00002,0x021bcb3110c00003,0x021bcb3110c00004,0x021bcb3110c00005,0x021bcb3110c00006,0x021bcb3110c00007,0
elect name,enum from dbcolumns w where tablename = [photoall[and enum !
elect name,enum from dbcolumns where tablen where tablename = [photoall[and enum !
select company where email = ydrh@gmail.com
select db where id = 63gfg74xcv
select details where email = 63gfg74xcv,63gfg74xcv
select * reg where email = fggff@gmail.com
select * from field where fieldid = 0x081505b681b40000
select * from photoobjectance where objid = 0x081505b681b40102,0x081505b681b40201,0x081505b681b40202,0x081505b681b40203,0x081505b681b40204,0x081505b681b40205,0x081505b681b40206,0x081505b681b402
select * from photoz where objid = 0x0815054161c104c8,0x081505b681b40102,0x0815054161c104c8,0x0815054161c105c8,0x0815054161c105c9,0x0815054161c105c10,0x0815054161c105c11,0x0815054161c105c12,0x08
select * from speclineindex where specobjid = 0x021bcb3110c00000,0x021bcb3110c00011,0x021bcb3110c0012,0x021bcb3110c00001,0x021bcb3110c00002,0x021bcb3110c00003,0x021bcb3110c00004,0x021bcb3110c00005,0x021bcb3110c00006,0x021bcb3110c00007,0
select fname where id = 656545,65ghs6545,65ghs6545
select fname,lname where id = 65ghs6545
select name,enum from dbcolumns where tablename = [photoall[and enum !

Fig 7:- Output

Result shows the difference I query count after finding similarity metrics

ISSN(Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 5, May 2019

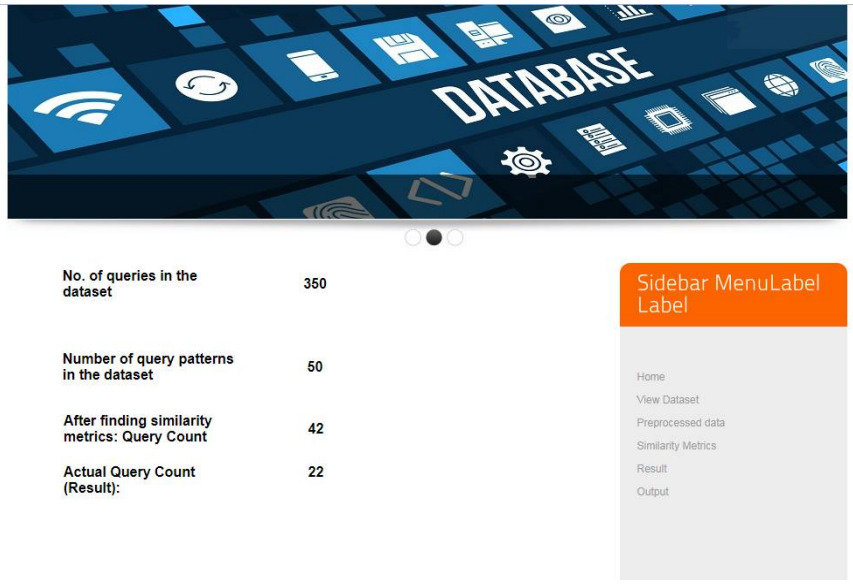


Fig 8:-Result

V. CONCLUSION

The focus of work is to know and developed similarity metrics for SQL queries relying on query structure to be used to cluster queries. Description of a quality assessment scheme that imprison the idea of query task using student answers to query-construction problems and a real-world Smartphone query load. Used of this system is to evaluate three existing query similarity metrics. Suggest a feature engineering method for standardizing query representations. Through more experiments, showed that unusual workloads have different characteristics and no one similarity metric reviewed was always good. The feature engineering steps present a development across the board since they addressed the error reasons recognized. The move toward described only represents the first steps to tools for summarizing logs by tasks.

ACKOWLEDGMENT

I would like to thank our honorable Principal, Dr. R. P. Singh and my special thanks to my guide and Head of Department Prof. Dinesh. D. Patil,& sincere thanks to all the respected teaching faculties of Department of Computer Science & Engineering of Hindi Seva Mandal's, Shri Sant Gadge Baba College of Engineering and Technology, Bhusawal. My special gratitude to all the authors of reference paper that are referred by me.

REFERENCES

[1] Gokhan Kul, Duc Thanh Anh Luong, Ting Xie, Varun Chandola, Oliver Kennedy, and Shambhu Upadhyaya, "Similarity Metrics for SQL Query Clustering", in IEEE 2018.
[2] N. Bruno and S. Chaudhuri, "Automatic physical database tuning: A relaxation-based approach," in ACM SIGMOD, 2005.
[3] O. Kennedy, J. Ajay, G. Challen, and L. Ziarek, "Pocket Data: The need for TPC-MOBILE," in TPC-TC, 2015.
[4] G. Kul, D. Luong, T. Xie, P. Coonan, V. Chandola, O. Kennedy, and S. Upadhyaya, "Ettu: Analyzing query intents in corporate databases," in WWW Companion, 2016.
[5] C. Dwork, "Differential privacy," in Automata, Languages and Programming, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds. Springer, 2006.
[6] C. Sapia, "Promise: Predicting query behavior to enable predictive aching strategies for OLAP systems," in DaWaK, 2000.
[7] A. Giacometti, P. Marcel, E. Negre, and A. Soulet, "Query recommendations for OLAP discovery driven analysis," in ACM DOLAP, 2009.
[8] X. Yang, C. M. Procopiuc, and D. Srivastava, "Recommending join queries via query log analysis", in IEEE ICDE, 2009.
[9] K. Stefanidis, M. Drosou, and E. Pitoura, ""You May Also Like" results in relational databases," in ACM DOLAP, 2009.

ISSN(Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 5, May 2019

[10] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu, "Snip- Suggest: context-aware autocompletion for SQL," pVLDB, 2010.
[11] G. Chatzopoulou, M. Eirinaki, S. Koshy, S. Mittal, N. Polyzotis, and J. S. V. Varman, "The QueRIE system for personalized query recommendations," IEEE Data Eng. Bull., 2011.
[12] W. Gatterbauer, "Databases will visualize queries too," pVLDB, 2011.
[13] J. Aligon, K. Boulil, P. Marcel, and V. Peralta, "A holistic approach to OLAP sessions composition: The falseto experience," in ACM DOLAP, 2014.
[14]K. Aouiche, P.-E. Jouve, and J. Darmont, "Clustering-based materialized view selection in data warehouses," in ADBIS, 2006.
[15]J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, and E. Turricchia, "Similarity measures for OLAP sessions," Knowledge and information systems, 2014.
[16] V. H. Makiyama, M. J. Raddick, and R. D. Santos, "Text mining applied to SQL queries: A case study for the SDSS SkyServer," in SIMBig, 2015.
[17] B. Chandra, B. Chawda, B. Kar, K. V. Reddy, S. Shah, and S. Sudarshan, "Data generation for testing and grading SQL queries," VLDBj, 2015.
[18] M. J. Zaki and W. Meira Jr, Data mining and analysis: fundamental concepts and algorithms. Cambridge University Press, 2014.
[19]A. Kamra, E. Terzi, and E. Bertino, "Detecting anomalous access patterns in relational databases," VLDBJ, 2007.
[20] S. Mathew, M. Petropoulos, H. Q. Ngo, and S. Upadhyaya, "A data-centric approach to insider attack detection in database systems," in RAID, 2010.
[21] H. V. Nguyen, K. B¨ohm, F. Becker, B. Goldman, G. Hinkel, and E. M¨ uller, "Identifying user interests within the data space-a case study with skyserver," in EDBT, 2015.
[22]R. Agrawal, R. Rantzau, and E. Terzi, "Context-sensitive ranking," in ACM SIGMOD, 2006.
[23] A. K. Chandra and P. M. Merlin, "Optimal implementation of conjunctive queries in relational data bases," in STOC, 1977.
[24] B. Chandra, M. Joseph, B. Radhakrishnan, S. Acharya, and S. Sudarshan, Partial marking for automated grading of SQL queries," VLDBj, vol. 9, no. 13, pp. 1541–1544, Sep. 2016.
[25]C. Sapia, "On modeling and predicting query behavior in OLAP systems." in DMDW, 1999.
[26]P. Seshadri, H. Pirahesh, and T. Y. C. Leung, "Complex query decorrelation," in IEEE ICDE, 1996.
[27]A. Gupta, V. Harinarayan, and D. Quass, "Aggregate-query processing in data warehousing environments," in pVLDB, 1995.
[28] F. Li and H. V. Jagadish, "Constructing an interactive natural language interface for relational databases," pVLDB, 2014.
[29]N. Shervashidze, P. Schweitzer, E. J. V. Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," JMLR, 2011.
[30]Miss. Pooja Subhash Joshi, and Prof. D. D. Patil, "A Review Paper on Correspondence Methods for SQL Query Clustering", International Journal of Research in Advent Technology ,Special Issue, Convergence 2019, pg.26-29.