Turkish Journal of Electrical Engineering and Computer Sciences

Volume 28 | Number 4 Article 27

1-1-2020

BIBSQLQC: Brown infomax boosted SQL query clustering algorithm to detectanti-patterns in the query log

VINOTHSARAVANAN RAMAKRISHNAN

PALANISAMY CHENNIAPPAN

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 2200 – 2212
© TÜBİTAK
doi:10.3906/elk-1912-108

BIBSQLQC: Brown infomax boosted SQL query clustering algorithm to detect anti-patterns in the query log

Vinothsaravanan RAMAKRISHNAN[*], **Chenniappan PALANISAMY**
Department of Information Technology, Bannari Amman Institute of Technology, Erode, India

Received: 18.12.2019 • **Accepted/Published Online:** 08.04.2020 • **Final Version:** 29.07.2020

Abstract: Discovery of antipatterns from arbitrary SQL query log depends on the static code analysis used to enhance the quality and performance of software applications. The existence of antipatterns reduces the quality and leads to redundant SQL statements. SQL log includes a large load on the database and it is difficult for an analyst to extract large patterns in a minimal time. Existing techniques which discover antipatterns in SQL query face a lot of innumerable challenges to discover the normal sequences of queries within the log. In order to discover the antipatterns in the log, an efficient technique called Brown infomax boosted SQL query clustering (BIBSQLQC) technique is introduced. Initially, the number of patterns (i.e. queries) are extracted from the SQL query log. After extracting the patterns, the ensemble clustering process is carried out to find out the antipatterns from the given query log. The Brown infomax boost clustering is an ensemble learning method for grouping the patterns by constructing several weak learners. The Brown clustering is used as a weak learner for partitioning the patterns into 'k' number of clusters based on the Euclidean distance measure. Then the weak learner merges the two clusters with maximum information gained to minimize the time complexity. The clustering results of weak learners are combined into strong results with minimal error rate (ER). By this way, the antipattern in the SQL query log is detected with a higher accuracy. Experimental evaluation is conducted with different parameters namely detection accuracy (DA), false positive rate (FPR) and time complexity (TC) using the two SQL query log data-sets (DS). The experimental result shows that, the BIBSQLQC technique achieves higher DA with lower TC and FPR than the conventional methods.

Key words: SQL log analysis, patterns and antipatterns, Brown infomax boosting, query, clustering

1. Introduction

A structured query language (SQL) is a database query language used for creating and managing the database by employing create, insert, delete, modify and update. In order to extract the intended information from multiple relations, the SQL provides the functions and operations like union, intersection and minus based on the concept defined in the set theory and relational algebra. Recently, different databases from various scientific domains are broadly available in public.

The SQL query log comprises of various patterns. SQL log records are the initial levels for database administration from the database performance tuning to benchmark design, etc. The query logs are enormous and it is challenging to an analyst to extract the patterns from a set of queries. Clustering is an essential process used for understanding the large query logs. It is the method of partitioning the set of patterns into clusters, where patterns in the same cluster tend to be more similar to one another than the patterns in different clusters based on predefined criteria.

[*]Correspondence: rvinothsaravanan@bitsathy.ac.in

Query clustering is a class of methods employed for grouping the patterns that are semantically related in query repository. The development of query clustering methods comes recently from contemporary web searching practice. The 3 primary applications of query clustering [1] are the identification of frequently asked questions, index term selection and query reformulation. The query clustering is divided into 5 types, namely concept-based query clustering, content-based query clustering, personalized concept-based query clustering, session-based query clustering and graph-based query clustering.

With a large amount of information stored in relational databases, it is essential to write the SQL queries to execute in a faster way. A pattern is a sequence of SQL query that points to certain functionality. Anti-patterns in SQL are the common errors which need to be avoided to perform the query processing at a faster rate. Antipattern detection is important in the context of SQL query processing. The antipatterns (i.e. error) are the patterns, which typically direct to redundant SQL statements and in many cases introduce more problems. For refactoring and postprocessing, discovery of antipatterns in the log is essential. Thus, such solution typically directs to the application scalability, testability and maintenance. Several techniques are introduced for discovering the antipatterns, they are complex and time-consuming. Therefore, to discover patterns and antipatterns in random SQL query logs, the clustering technique is employed.

The major contribution of the proposed BIBSQLQC technique is summarized as follows:

- To improve the antipattern DA in SQL query log, BIBSQLQC technique is developed. Using distance measure, the brown clustering is applied to separate the number of patterns into 'k' different clusters. The squared Euclidean distance is determined between patterns and cluster mean. Thus, the similar queries are grouped into a particular cluster. After that, a boosting technique combines each weak learner's results and creates strong clustering results by using gradient ascent function. The gradient ascent function is used for identifying the maximum mutual dependence between the two clusters. Hence, the strong clustering result effectively detects the antipatterns in the SQL query log.

- BIBSQLQC technique discovers the weak learner with a minimal training error to reduce the FPR. Here, the boosting technique combines weak learners and assigns similar weights. Then, the training error is computed for each weak learner where the boosting algorithm updates the weight of the weak learner based on the error value. If the weak learners are correctly grouped, there is a decrease in weight value. Otherwise, the weight value is increased and the patterns are wrongly grouped. Finally, the strong clustering results are obtained with minimum ER.

- To minimize the TC, the maximum information gains between the 2 clusters are computed in the proposed BIBSQLQC technique. The maximum mutual dependence between the clusters are merged into one single cluster. Then the boosting technique effectively identifies the patterns and antipatterns with minimum time.

2. Related works

A flexible method was developed in [2] with the help of tools to identify the antipatterns from the SQL queries log. This method is able to detect the antipattern with better accuracy whereas the time taken for detection is too long. A new online detection technique was introduced in [3] for finding the SQL injection attack from the query log. This technique achieved higher accuracy rate for identifying the SQL injection attack but the performance of time remained unsolved.

To discover SQL injection attacks from the database with higher accuracy, a support vector machine classification and different kernel functions were designed [4]. But the ER was not minimized during the classification. A new approach was introduced [5] to identify the injection attacks by using the SQL and support vector machine (SVM). But this also failed as it was not able to focus on improving the accuracy of attack detection.

Later, a hot query bank (HQB) was designed [6] for improving the efficiency of SQL injection attack detection. Though the approach minimized the execution time, the ER was not minimized. Hence a gap-weighted string subsequence kernel algorithm was developed [7] to detect the subsequences of query strings and categorize indefinite test queries using support vector machine which resulted in failure in minimizing the error of the classification. Finally clustering-based fragmentation method was developed [8] for solving the data replication problem from the SQL query but, TC of the clustering remained unsolved.

Identification of complex periodic patterns in the SQL query database was performed [9] with minimum time consumption. However, this approach also failed to discover the antipatterns based on clustering process. A cost-based query optimization approach was developed in [10] for selecting and grouping the complex SQL queries. But, the designed optimization approach failed to minimize the TC while processing the complex query optimization.

A query-refinement method was developed [11] for answering the top-k query in the context of SQL. This method minimized the run-time, but failed to improve the quality while using the nonnumeric attributes. Then, a logical framework was introduced [12] to evaluate the complex subqueries using SQL. But here also the performance did not improve while processing the complex subqueries in distributed environments.

A cyclic graph based approach was developed [13] for optimizing the linear recursive queries in SQL, but it failed to detect the duplicate queries in the SQL with higher accuracy. An SVM based ranking method was introduced [14] to rank the SQL query results, which functioned with limited and heterogeneous users preferences.

A start end mid algorithm and web-based SQL compiler were designed [15] for detecting errors. Though the algorithm increased the accuracy, the TC in the error detection was not minimized. An intrusion detection based on a multi-agent architecture was developed [16] to identify and block the SQL injection. The designed approach minimized the execution time, but still the accurate identification was not completely successful.

A new fault localization method was introduced [17] to discover the faults in SQL queries, which resulted in poor efficiency of fault detection. An integrated approach was developed [18] to process the SQL query with minimal time. This approach did not enhance the query processing performance. A feature engineering strategy was developed in [19] for clustering the number of SQL queries using different similarity metrics. The strategy failed to improve the clustering quality with minimum error rate. A pattern detection method was introduced in [20] for identifying the patterns and antipatterns from the query log of a database. The designed method failed to improve antipattern detection accuracy and minimizes the time complexity. A cluster-based decision tree approach was introduced [21] to identify the most relevant patterns and syntax error from the database. This approach failed to analyze more patterns with minimum complexity.

A novel SQL injection detection method based on neural network was presented in [22]. However, time complexity was higher. In [23], a novel cost model was introduced for Spark SQL. But, the minimization of error was not enough. A novel data partitioning scheme was designed in [24] for distributing data over database clusters. But, the performance of accuracy was not considered.

Therefore, from the existing survey it was found out that conventional techniques had a few limitations such as lack of improving the antipattern DA, more TC, high FPR, and so on. Thus, such kinds of issues are addressed by introducing a novel clustering technique called, BIBSQLQC technique to improve the antipattern DA.

This paper is organized into 6 different sections. Section 2 discusses the reviews of the related works with their limitations of current performances. Section 3 provides a detailed explanation of our proposed BIBSQLQC technique. Section 4 presents the Experimental evaluation of BIBSQLQC technique and compared with the existing methods. Section 5 provides the results and discussion of the parameters. Conclusion is given in Section 6.

3. Methodology

Discovering the antipatterns from the SQL query log is beneficial where the developer performs refactoring in the source code. This helps to improve the software quality. Therefore, a BIBSQLQC technique is presented to identify the patterns and antipatterns in arbitrary SQL query logs. The processing diagram of BIBSQLQC technique is portrayed in Figure 1. to identify the patterns and antipatterns from SQL query log.

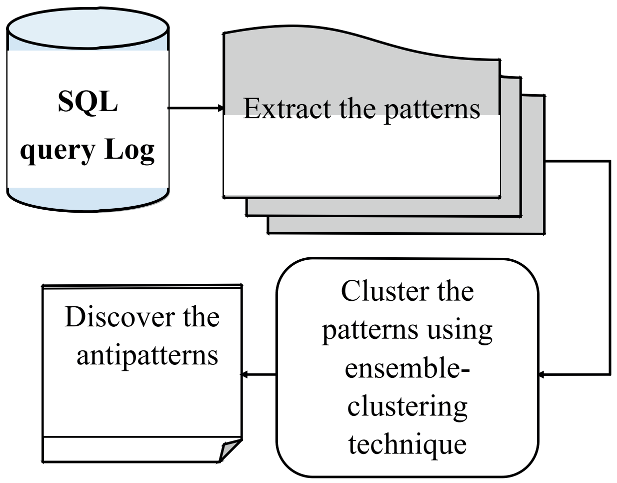
SQL query Log
Extract the patterns
Cluster the patterns using ensemble-clustering technique
Discover the antipatterns

Figure 1. Flow process of proposed BIBSQLQC technique.

3.1. Brown infomax boosted SQL query clustering technique

Brown infomax boost clustering is a machine learning ensemble meta-algorithm which changes the output of weak learners into strong ones. A weak learner is a base cluster which provides slightly correlated results and a strong learner is a cluster which is arbitrarily well correlated with the true output results. The strong cluster performs better than any of the simple clusters with randomly small error probability than the individual clustering results. Therefore, the proposed BIBSQLQC technique uses the ensemble technique to improve clustering performance by combining the results of weak clusters. The proposed technique uses the brown clustering, whereas a weak learner finds the antipatterns in the given SQL query log. Clustering is the process of grouping similar SQL query patterns into clusters. The patterns (i.e. a sequence of queries) within a group are similar for increasing the antipatterns DA. The ensemble process of BIBSQLQC technique is described as follows.

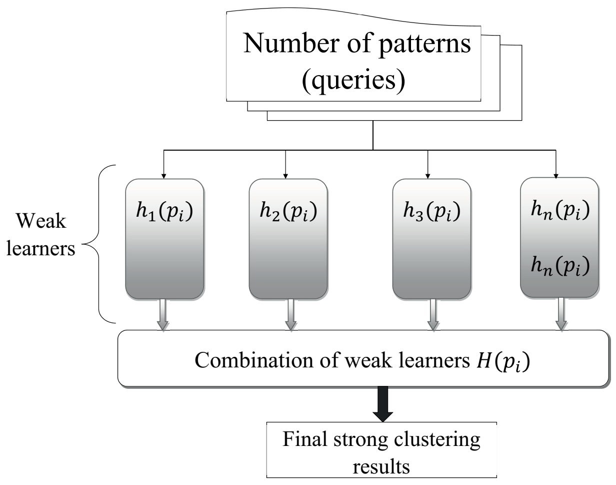
Number of patterns (queries)
Weak learners
$h_1(p_i)$
$h_2(p_i)$
$h_3(p_i)$
$h_n(p_i)$
$h_n(p_i)$
Combination of weak learners $H(p_i)$
Final strong clustering results

Figure 2. Processing of Brown infomax boost clustering.

The flow process of Brown infomax boost clustering is portrayed in Figure 2. Let us assume 'n' number of patterns (i.e. queries) from SQL query log $P_i, i = 1, 2, 3...n$ and are partitioned into 'k' number of clusters C_1, C_2, C_3, C_k, which in turn form a strong cluster $H(p_i)$, by combining the number of weak learners $h_1(p_i), h_2(p_i), h_3(p_i), h_n(p_i)$. The brown clusters act as a weak learner to partition the entire patterns into different clusters. The brown clustering is a hierarchical clustering algorithm used to merge two clusters as one and move up in the hierarchy. Initially, the brown clustering algorithm initializes the 'k' number of clusters. For each cluster, the mean value is assigned to partition the patterns.

In the hierarchical clustering algorithm, the clustering is performed using a distance metric. The distance between the mean and the patterns are computed to find out the antipatterns from the SQL query log. Thus the squared Euclidean distance is calculated between the mean of the cluster and the patterns to find out similar queries which are calculated as follows:

$$D(C_j, P_i) = \sum_{j=1}^{k} \sum_{i=1}^{n} (C_j - P_i)^2. \tag{1}$$

In (1), D denotes a distance, C_j denotes a mean of the cluster, P_i denotes patterns in the SQL query log. The minimum distance among the mean and patterns are grouped into a particular cluster.

$$arg\ min\ D(C_j, P_i) \tag{2}$$

In (2), *arg min* indicates an argument of a minimum function to discover the minimum distance between the pattern and mean of the cluster. By this way, all the weak learners identify similar patterns and antipatterns. In the next process, the mutual information between the clusters are calculated to merge the two clusters for minimizing the TC in pattern detection. The mutual information determines the mutual dependence between the two clusters. The probabilities of mutual dependence between the clusters are computed as follows:

$$MD(C_1, C_2) = p(C_1, C_2)\ \log_2 \left(\frac{p(C_1, C_2)}{p(C_1)\ p(C_2)} \right). \tag{3}$$

In (3) $MD(C_1, C_2)$ denotes the mutual dependence between the 2 clusters C_1, C_2, $p(C_1, C_2)$ represent the joint probability distribution and $p(C_1)$, $p(C_2)$ represent the marginal probability between the 2 clusters.

By using (3), the mutual dependence between the 2 clusters is computed. Secondly, the gradient ascent function is used for finding the maximum dependence between the 2 clusters.

$$F(x) = arg\ max\ MD(C_1, C_2). \tag{4}$$

In (4) $F(x)$ denotes a gradient ascent function, $arg\ max$ denotes an argument of the maximum function, $MD(C_1, C_2)$ denotes the mutual dependence between the 2 clusters. Then the merging operation is performed by combining the 2 clusters C_1, C_2. As a result, the maximum mutual dependence between the clusters is merged. Finally, the 2 clusters are obtained as similar patterns and antipatterns. The results of the weak learner do not improve the clustering performance but show some training error. Hence the weak learners are boosted to achieve higher clustering accuracy. In order to obtain the strong learner, the weak learners are combined as follows:

$$H(p_i) = \textstyle\sum_{t=1}^{n} h_i(p_i). \tag{5}$$

In (5),$H(p_i)$ indicates a strong cluster and $h_i(p_i)$ signifies a weak learner output whereas, the similar weight value is allocated to each weak cluster.

$$H(p_i) = \textstyle\sum_{t=1}^{n} \delta\ h_i(p_i). \tag{6}$$

In (6),δ denotes a weight of the weak learners $h_i(p_i)$. The weight is a random integer. Followed by this, the training error is calculated for each weak learner. The error is calculated as a squared difference between the actual and predicted output of the weak learners.

$$\alpha_E = (Y_i - H(p_i))^2. \tag{7}$$

In(7), α_E denotes a training error, Y denotes an actual output and $H(p_i)$ represents a predicted output. Based on the error, the weights are readjusted and this is called as reweighting. The wrongly grouped patterns gain a higher weight. If the weak learners are correctly grouped, there is a decrease in weight value. Thus, strong clustering results are attained to choose the weak learner with a lower training error. The obtained strong results increase the clustering accuracy (CA) and reduce FPR. Algorithmic process of BIBSQLQC technique is described as follows.

Algorithm 1 describes the brown infomax boosted clustering to identify the antipatterns in the given SQL query log. Initially, the number of patterns are collected from the log and applied to the ensemble clustering technique which construct the multiple weak learners into cluster patterns. The brown hierarchical clustering is applied for grouping the patterns into different clusters. The clustering is performed with a minimum distance between the cluster's mean and patterns. Therefore the weak learner merges the two clusters with maximum information gain (i.e. mutual dependence). Finally, the ensemble technique combines all the weak learners and creates a strong one to increase the clustering performance. The similar weight is assigned to each weak learner and it calculates the training error. Then the weight is updated based on the error value. The weak learner with the minimum error is the stopping criterion. Based on the clustering results, the antipatterns in the given SQL query log are identified with high accuracy.

Algorithm 1: Brown infomax boosted SQL query clustering

```
    Input: SQL query log, number of patterns (i.e. queries) $P_i, i = 1, 2, 3..n$
    Output: Improve antipattern detection accuracy
1   begin
2   for all $P_i, i = 1, 2, 3..n$ do
3       Construct the ‘n’ weak learners $h_i(p_i)$
4       Define ‘k’ number of clusters and mean $C_j$
5       Compute squared Euclidean distance $D(C_j, P_i)$
6       Group patterns to cluster $C_j$ with minimum distance $argminD(C_j, P_i)$
7       Calculate the mutual dependence between the 2 clusters $MD(C_1, C_2)$
8       Merge 2 clusters with maximum dependence $arg\ max\ MD(C_1, C_2)$
9       Obtain the 2 clusters similar patterns and antipatterns
10      Combine all the weak learner's results $\sum_{t=1}^{n} h_i(p_i)$
11      for each weak learner $h_i(p_i)$ do
12          Initialize a similar weight $\sum_{t=1}^{n} \delta\ h_i(p_i)$
13          Calculate the training error $\alpha_E$
14          Update the weight $\delta'$
15          Select weak learner with minimum error $arg\ min\ \alpha_E\{h_i(p_i)\}$
16          Obtain strong clustering results
17      end
18  end
19  end
```

4. Experimental evaluation

Experimental evaluation of BIBSQLQC technique and the existing methods namely feature engineering technique [19], pattern detection method [20] and clustering based decision tree approach [21] are implemented using Java language. The SQL query log such as IIT Bombay DS and UB Exam DS are used for conducting the experiments. The SQL queries are collected from a repository hosted in GitHub. [1]

IIT Bombay DS comprises of students' responses to SQL questions specified in IIT Bombay's undergraduate databases course. DS comprises students' answers to 14 diverse query writing tasks. IIT Bombay DS is employed for answers to homework assignments. Whereas the DS comprises of 3 columns namely ID, Label and query. The IIT Bombay DS comprises 630 instances. Based on the stored information in the SQL query log, first, similar sequences of queries within the log are identified through the clustering process. The antipatterns and similar patterns are identified based on the clustering results.

UB Exam DS includes the students' answers to SQL questions and specified as part of the department's graduate database course. To cluster such similar queries and antipatterns, DS includes various query files. It includes queries with 2 years of midterm exams of 2014 and 2015, respectively. Different queries gathered from DS and antipatterns within the SQL query log are identified with a higher accuracy.

Totally 10 different runs are performed using 2 different DS with a number of queries. The performance of BIBSQLQC technique and existing methods namely, feature engineering technique [19], pattern detection method [20] and clustering based decision tree approach [21] are evaluated with certain metrics such as DA, FPR and TC.

[1]Microsoft (2018). Github [online]. Website https://Github.com/UBOdin/EttuBench/tree/master/data/ [accessed 21 January 2019].

5. Results and discussions

In this section, the experimental results of the proposed BIBSQLQC technique and existing methods [19–21] are discussed with different metrics such as DA, FPR and TC. Performances of all the 3 methods are estimated and the results are reported in the Table and a 2-dimensional graph.

5.1. Performance analysis of detection accuracy

DA is measured as a number of patterns (i.e. queries) and antipatterns are correctly identified through the clustering process to the total number of patterns. DA is formalized as below:

$$Detection\ accuracy = \frac{Number\ of\ queries\ correctly\ detected}{n} * 100. \tag{8}$$

In (8), n indicates a number of patterns. DA is measured in percentage (%). The mathematical calculation of the normal and antipattern DA using two DS is given below.

The DA of normal and antipattern is portrayed in Tables 1 and 2 with IIT Bombay DS and UB exam DS. Tables 1 and 2 clearly show DA for 3 different methods namely BIBSQLQC and the existing methods [19–21] with the number of queries (i.e. patterns). From Tables 1 and 2, it is evident that the DA is improved using the BIBSQLQC technique. This significant improvement is achieved using ensemble clustering technique.

Table 1. Detection accuracy versus number of queries using IIT Bombay dataset.

Number of queries	Detection accuracy (%)			
	BIBSQLQC	Feature engineering technique	Pattern detection method	Clustering based decision tree approach
50	94	88	84	80
100	92	87	83	77
150	90	86	82	75
200	89	85	81	73
250	88	84	80	71
300	87	83	79	69
350	86	82	78	67
400	88	80	76	65
450	85	79	75	63
500	84	78	74	60

The IIT Bombay DS is used for conducting the experiments with a number of user queries ranging from 50 to 500. Eventually 10 different runs are performed to obtain accuracy detection. The results of the proposed BIBSQLQC technique is compared with the accuracy results of the existing techniques. The BIBSQLQC technique enhances the normal and abnormal DA by 6%, 11% and 26% as compared to the existing one [19–21].

The UB exam DS is applied for calculating the DA with the number of queries ranging from 20 to 200. The different accuracy results are obtained using techniques like BIBSQLQC, feature engineering technique [19], pattern detection method [20] and clustering based decision tree approach [21]. As shown in Table 2, it is considered that the number of queries are 20. They are clustered into similar or antipatterns. Out of these 20, 19 are correctly clustered and their accuracy is 95% when BIBSQLQC technique is used. But the

Table 2. Detection accuracy versus number of queries using UB exam dataset.

Number of queries	Detection accuracy (%)			
	BIBSQLQC	Feature engineering technique	Pattern detection method	Clustering based decision tree approach
20	95	90	85	82
40	93	88	83	79
60	90	87	82	77
80	89	86	81	75
100	88	85	80	73
120	87	84	79	70
140	86	81	78	69
160	85	80	77	67
180	84	79	76	65
200	83	78	75	62

conventional techniques achieve 90%, 85% and 82% of accuracy. Similarly, 9 remaining runs are performed and various results are obtained. The reported results clearly show that the DA of similar and antipatterns are significantly improved by 6%, 12% and 24% than the conventional feature engineering technique [19], pattern detection method [20] and clustering based decision tree approach [21].

5.2. Performance analysis of the false positive rate

FPR is defined as the number of patterns and antipatterns incorrectly identified through the clustering process to the total number of patterns (i.e. queries). FPR is expressed as follows:

$$FPR = \frac{Number\ of\ queries\ incorrectly\ clustered}{n} * 100. \tag{9}$$

In (9),n denotes a number of patterns. FPR is measured in percentage (%).

The performance result of FPR is portrayed in Figures 3 and 4 with IIT Bombay DS and UB exam DS. From Figures 3 and 4, the number of queries are taken as input in the x axis whereas FPR are obtained in y axis. From Figures 3 and 4, the violet curve points to FPR of BIBSQLQC technique whereas red and green curves point to FPR of the existing one [19–21]. Thus, the graphical result confirms that the FPR is considerably minimized using BIBSQLQC technique when compared to the conventional techniques. This is because BIBSQLQC technique uses the ensemble technique to distinguish the similar patterns and antipatterns with different clusters. The similar queries are grouped after collecting the data with the assistance of brown clustering. For dividing the similar sequence of queries and dissimilar sequence of queries, the Euclidean distance is employed to measure the distance between SQL patterns and cluster mean. The boosting technique combines all base clusters and assigns a similar weight. The training error for each cluster is calculated. The weight of the base cluster is adjusted based on the error value. Finally, the strong clustering results are obtained with minimum ER which assists to reduce the FPR.

Using IIT Bombay DS, the FPR of BIBSQLQC technique is reduced by 29%, 42% and 53% as compared to the existing [19–21] . Additionally, the BIBSQLQC technique minimizes the FPR by 30%, 46% and 51%

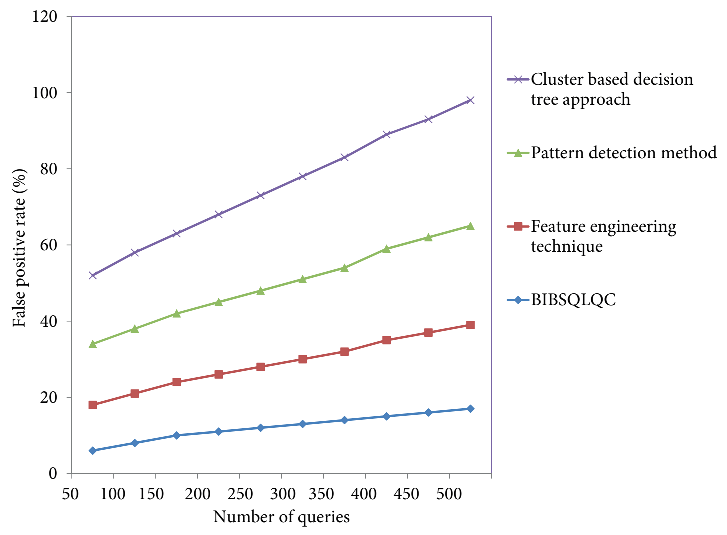

Figure 3. Performance results of false positive rate versus number of queries using IIT Bombay dataset.

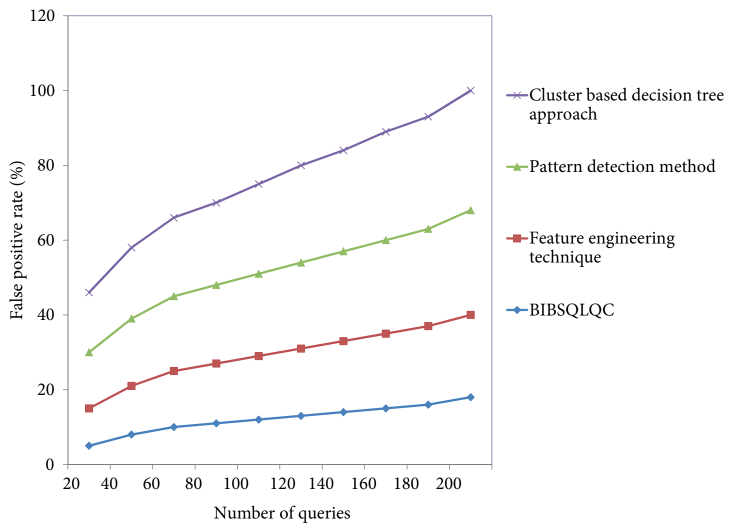

Figure 4. Performance results of false positive rate versus number of queries using UB Exam dataset.

than the conventional feature engineering technique [19], pattern detection method [20] and clustering based decision tree approach [21].

5.3. Performance analysis of time complexity

TC is measured as the amount of time consumed to identify patterns and antipatterns through the clustering process. The formula for calculating the TC is given below.

$$TC = Number\ of\ patterns * time(identify\ one\ query). \tag{10}$$

From (10), TC denotes a time complexity. TC is measured in milliseconds (ms).

The performance of TC is portrayed in Tables 3 and 4 with the number of queries (i.e. patterns). The results of different TC are obtained for various set of queries which is taken from IIT Bombay ranging from 50

Table 3. Time complexity versus the number of queries using IIT Bombay dataset.

Number of queries	Time complexity (ms)			
	BIBSQLQC	Feature engineering technique	Pattern detection method	Clustering based decision tree approach
50	13	15	18	19
100	16	19	22	24
150	20	24	27	29
200	24	28	32	35
250	27	30	35	37
300	31	35	39	41
350	35	40	44	45
400	37	42	46	47
450	43	45	50	53
500	49	52	55	57

Table 4. Time complexity versus the number of queries using UB Exam dataset.

Number of queries	Time complexity (ms)			
	BIBSQLQC	Feature engineering technique	Pattern detection method	Clustering based decision tree approach
20	10	12	14	16
40	12	14	17	20
60	14	18	21	25
80	18	22	25	30
100	22	25	30	34
120	25	28	32	38
140	27	29	35	40
160	29	34	38	44
180	31	36	40	47
200	33	38	42	50

to 500 and from UB exam DS between 20 and 200. The results show that the proposed BIBSQLQC technique minimizes the TC when compared with the other techniques, as it is implemented based on the boosting technique which considers the brown clustering as a weak learner. The weak learners initialize the number of clusters and mean values, whereas the Euclidean distance between the mean and queries are computed to find out similar queries. To reduce the antipattern detection time, the maximum information gained from the 2 clusters are combined into a single cluster. Eventually, the boost clustering algorithm detects the patterns and antipatterns by combining all the base clusters. This process helps to minimize the TC.

The TC of BIBSQLQC technique is only 13 ms from the 50 patterns taken from the IIT Bombay DS, whereas it is 15 ms, 18 ms and 19 ms by the existing method [19–21]. This proves that the considerable improvements can be made by the use of BIBSQLQC technique. Finally in the process of comparison, the results show that the BIBSQLQC technique reduces the TC by 12%, 21% and 25% when compared to the existing methods.

For UB exam DS, the TC of BIBSQLQC technique, the existing [19–21] are 10 ms, 12 ms, 14 ms and 16 ms while considering the 20 number of input queries. Similarly, from various number of input queries, 9 different results are obtained. The comparison of 10 different results show that the TC is considerably minimized by 14%, 26% and 36% using the BIBSQLQC technique when compared with the conventional methods.

The above results and discussion prove that the antipatterns are properly identified from the SQL query log with a higher accuracy, minimal TC and FPR.

6. Conclusions

The BIBSQLQC technique is introduced to enhance the antipatterns of DA with less TC. The accurate detection is obtained by grouping the similar sequence of queries into different local clusters. The boosting technique uses the brown hierarchical clustering algorithm for dividing the number of SQL queries into different local clusters through the Euclidean distance measure. The minimal distance between the clusters mean and queries are grouped into a particular cluster. The maximum dependence between the clusters are merged to handle a large number of clusters which in turn reduce the TC in antipattern detection. Later, the weak clustering results are grouped together to obtain strong clustering results. The boosting technique increases the CA with minimum ER. This ensemble technique increases the antipattern DA and minimizes the FPR. The experimental evaluation is carried out with 2 DSs and certain parameters namely DA, FPR and TC. Thus from the performance results and discussions, the BIBSQLQC technique yields a higher antipattern DA with minimum FPR and TC when compared with the other conventional methods.

References

[1] Wen JR, Zhang HJ. Query clustering in the web context. In: Xiong H (editor). Clustering and Information Retrieval. Springer, USA: Springer Press, 2004, pp. 195-225

[2] Sabir F, Rasool G, Yousaf M. A Lightweight approach for specification and detection of SOAP Anti-Patterns. International Journal of Advanced Computer Science and Applications 2017; 8 (5): 455-467. doi: 10.14569/IJACSA.2017.080555

[3] Badia A, Wagner A. Complex SQL Predicates as Quantifiers. IEEE Transactions on Knowledge and Data Engineering 2014; 26 (7): 1617-1630. doi: 10.1109/TKDE.2013.55

[4] Kim MY, Lee DH. Data-mining based SQL injection attack detection using internal query trees. Expert Systems with Applications 2014; 41 (11): 5416-5430. doi: 10.1016/j.eswa.2014.02.041

[5] Kar D, Panigrahi S, Sundararajan S. SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM. Computers & Security 2016; (60): 206-225. doi: 10.1016/j.cose.2016.04.005

[6] Chung YC, Wu MC, Chen YC, Chang WK. A Hot Query Bank approach to improve detection performance against SQL injection attacks. Computers & Security 2012; 31 (2): 233-248. doi: 10.1016/j.cose.2011.11.007

[7] McWhirter PR, Kifayat K, Shi Q, Askwith B. SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel. Journal of Information Security and Applications 2018; (40): 199-216. doi: 10.1016/j.jisa.2018.04.001

[8] Wiese L. Clustering-based fragmentation and data replication for flexible query answering in distributed databases. Journal of Cloud Computing 2014; 3 (18): 1-15. doi: 10.1186/s13677-014-0018-0

[9] Zimniak M, Getta JR, Benn W. Predicting database workloads through mining periodic patterns in database audit trails. Vietnam Journal of Computer Science 2015; 2 (4): 201-211. doi: 10.1007/s40595-015-0042-0

[10] JFan J, Zhang M, Kok S, Lu M, Ooi BC. Crowdop: Query optimization for declarative crowdsourcing systems. IEEE Transactions on Knowledge and Data Engineering 2015; 27 (8): 2078-2092. doi: 10.1109/TKDE.2015.2407353

[11] Jardiansah JT, Wibawa AP, Widyaningtyas T, Yasuhisa O. SQL logic error detection using start end mid algorithm. Knowledge Engineering and Data Science (KEDS) 2018; 1 (1): 33-38. doi: 10.17977/um018v1i12018p33-38

[12] Dalai AK, Jena SK. Neutralizing SQL injection attack using server side code modification in web applications. security and communication Networks 2017; (2017): 1-12. doi: 10.1155/2017/3825373

[13] Ordonez C. Optimization of linear recursive queries in SQL. IEEE Transactions on Knowledge and Data Engineering 2010; 22 (2): 264-277. doi: 10.1109/TKDE.2009.83

[14] Chen Z, Li T, Sun Y. A Learning approach to SQL Query results ranking using skyline and users' current navigational behavior. IEEE Transactions on Knowledge and Data Engineering 2013; 25 (12): 2683-2693. doi: 10.1109/TKDE.2012.128

[15] Xu W, He Z, Lo E, Chow CY. Explaining Missing Answers to Top-k SQL Queries. IEEE Transactions on Knowledge and Data Engineering 2016; 28 (8): 2071-2085. doi: 10.1109/TKDE.2016.2547398

[16] Pinzón CI, De Paz JF, Herrero Á, Corchado E, Bajo J et al. idMAS-SQL: intrusion detection based on MAS to detect and block SQL injection through data mining. Information Sciences 2013; (231): 15-31. doi: 10.1016/j.ins.2011.06.020

[17] Guo Y, Li N, Offutt J, Motro A. Exoneration-based fault localization for SQL predicates. Journal of Systems and Software 2019; (147): 230-245. doi: 10.1016/j.jss.2018.10.037

[18] Chandra B, Chawda B, Kar B, Reddy KVM, Shah S et al. Data generation for testing and grading SQL queries. The VLDB Journal 2015; 24 (6): 731-755. doi: 10.1007/s00778-015-0395-0

[19] Kul G, Luong DTA, Xie T, Chandola V, Kennedy O et al. Similarity metrics for SQL query clustering. IEEE Transactions On Knowledge And Data Engineering 2018; 30 (12): 2408-2420. doi: 10.1109/TKDE.2018.2831214

[20] Arzamasova N, Schaler M, Bohm K. Cleaning antipatterns in an SQL query log. IEEE Transactions On Knowledge and Data Engineering 2018; 30 (3): 421-434. doi: 10.1109/TKDE.2017.2772252

[21] Lino A, Rocha Á, Macedo L, Sizo A. Application of clustering-based decision tree approach in SQL query error database. Future Generation Computer Systems 2019; (93): 392-406. doi: 10.1016/j.future.2018.10.038

[22] Tang P, Qiu W, Huang Z, Lian H, Liu G. Detection of SQL injection based on artificial neural network. Knowledge-Based Systems 2020; 190: 105528. doi: 10.1016/j.knosys.2020.105528

[23] Baldacci L, Golfarelli M. A cost model for SPARK SQL. IEEE Transition on Knowledge Data Engineering 2019; 31(5): 819-832. doi: 10.1109/TKDE.2018.2850339

[24] Mrozek D, Kwiendacz J, Malysiak-Mrozek B. Protein construction-based data partitioning scheme for alignment of protein macromolecular structures through distributed querying in federated databases. IEEE Transaction on Nanobioscience. 2020;19 (1): 102-116. doi: 10.1109/TNB.2019.2930494