Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 2730246, 9 pages
https://doi.org/10.1155/2021/2730246

Research Article

Random Forest Bagging and X-Means Clustered Antipattern Detection from SQL Query Log for Accessing Secure Mobile Data

Rajesh Kumar Dhanaraj,[1] **Vinothsaravanan Ramakrishnan**,[2] **M. Poongodi**,[3] **Lalitha Krishnasamy**,[4] **Mounir Hamdi**,[3] **Ketan Kotecha**,[5] **and V. Vijayakumar**[6]

[1]*School of Computing Science and Engineering, Galgotias University, India*
[2]*Computer Science and Engineering, JAIN (Deemed-to-be-University), India*
[3]*College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar*
[4]*Department of Information Technology, Kongu Engineering College, India*
[5]*Symbiosis Centre for Applied Artificial Intelligence, Symbiosis International (Deemed University), Pune, India*
[6]*School of Computing Science and Engineering, University of South Wales, Sydney, Australia*

Correspondence should be addressed to Ketan Kotecha; drketankotecha@gmail.com

Received 13 September 2021; Accepted 8 November 2021; Published 24 November 2021

Academic Editor: Deepak Kumar Jain

In the current ongoing crisis, people mostly rely on **m**obile phones for all the activities, but query analysis and mobile data security are major issues. Several research works have been made on efficient detection of antipatterns for minimizing the complexity of query analysis. However, more focus needs to be given to the accuracy aspect. In addition, for grouping similar antipatterns, a clustering process was performed to eradicate the design errors. To address the above-said issues and further enhance the antipattern detection accuracy with minimum time and false positive rate, in this work, Random Forest Bagging X-means SQL Query Clustering (RFBXSQLQC) technique is proposed. Different patterns or queries are initially gathered from the input SQL query log, and bootstrap samples are created. Then, for each pattern, various weak clusters are constructed via X-means clustering and are utilized as the weak learner (clusters). During this process, the input patterns are categorized into different clusters. Using the Bayesian information criterion, the similarity measure is employed to evaluate the similarity between the patterns and cluster weight. Based on the similarity value, patterns are assigned to either relevant or irrelevant groups. The weak learner results are aggregated to form strong clusters, and, with the aid of voting, a majority vote is considered for designing strong clusters with minimum time. Experiments are conducted to evaluate the performance of the RFBXSQLQC technique using the IIT Bombay dataset using the metrics like antipattern detection accuracy, time complexity, false-positive rate, and computational overhead with respect to the differing number of queries. The results revealed that the RFBXSQLQC technique outperforms the existing algorithms by 19% with pattern detection accuracy, 34% minimized time complexity, 64% false-positive rate, and 31% in terms of computational overhead.

1. Introduction

In recent years, several databases from different domain areas, especially mobile databases, have been available in public for fast and precise accessibility. They usually provide interfaces and hence are said to be accessed extensively. However, due to the public availability of the database, the interaction between the owners and users is not said to occur. Also, analysing such a log is a cumbersome process, as given in [1].

Apart from the purpose of making a call, mobile phones are used for multipurposes like shopping, mobile banking, ticket booking, and social media applications. These devices can able to run a small business and help to maintain e-records. Where and how these data are stored? How to retrieve the data? And managing to handle the traffic generated by these data is cumbersome. Query analysis plays a vital role in retrieving the data both at the customer end and the developer or service provider. Pattern detection mechanisms are applied to detect the data,

but detection accuracy is another aspect to concentrate on. So, antipattern detection accuracy with minimum time complexity is required efficient query analysis in the query log dataset.

A cluster-based decision tree technique was proposed in [2] to identify the most relevant patterns. In this work, the clustering technique and decision trees were integrated to obtain meaningful patterns based on certain rule generation. Here, new knowledge standards were discovered based on computational intelligence techniques and source code analysis. Also, knowledge discovery in databases (KDD) and certain principles of lexical, syntactic, and semantic analysis were utilized to result in a minimum error [3].

In [4], extraction and analysis of patterns using the pattern detection technique were applied from a database's query log. Even though more concentration was made on minimizing the design errors (anti-patterns), which on the other hand, resulted in unnecessary SQL statements, however, those antipatterns did not have a negative influence. Therefore, efficient segregation between patterns and antipatterns was made. Yet another graph-based partitioning method called GPT was proposed in [5] using a novel graph-based database partitioning method. With this, the storage overhead was found to be minimized.

Thus, as given above, several techniques have been designed to address pattern clustering for significant antipattern detection. However, as provided above, the major challenge was attaining higher detection accuracy with minimum complexity and also the pattern detection from a large set of queries. In this work, the Random Forest Bagging X-means SQL Query Clustering (RFBXSQLQC) technique is proposed to address these issues. The major contributions of the RFBXSQLQC technique are given below:

(1) This work proposed a Random Forest Bagging X-means Clustering model to improve the antipattern detection accuracy with minimum time complexity during the user query clustering process

(2) Bayesian information criterion is proposed to enhance the relevant and irrelevant pattern clusters

(3) Designed a Random Forest Bagging-based Voting model to reduce both false positive rate and space complexity involved in the antipattern detection process

(4) Simulation and experimental results are proposed to be conducted to measure the antipattern detection accuracy, time complexity, false-positive rate, and space complexity for varied numbers of queries during antipattern detection

2. Related Works

In [6], distance-based clustering techniques were utilized for measuring similarity with respect to the Rand index. Here, both low and high-dimensional datasets were analyzed. Based on the analysis of similarity measures, the convergence rate was said to be improved and, hence, more advantageous for high dimensional. A Cooperative Parallel Evolutionary Algorithm (P-EA) was presented in [7] to detect antipatterns involving web service. With the utilization of the evolutionary algorithm, the computational complexity involved was said to be minimized with high precision.

Smart Aggregation of Antipattern Detectors (SMAD) were proposed in [8] using a machine learning-based ensemble method that collected different antipatterns. Moreover, a detection tool was also designed for higher detection accuracy. An improved version of XData was proposed in [9], utilizing the data generation method to maintain SQL queries and a superior class of mutations. The approach was provided better performance, but however, it failed in enhancing query processing performance. To address this issue, a systematic process that was considered by load testing and profiling data was presented in [10, 11] that utilized a software refactoring process to reduce the run time.

Several methods were utilized for detecting antipatterns which involved a large query log database. In [12], metric-based techniques were designed. Yet, another flexible method was proposed in [13], and also a new online detection method was proposed in [14] for improving the performance of antipattern detection from query log database. However, the accuracy factor was found to be compromised in certain cases. In [15], a detection schema for antipattern detection using the relational database was proposed to address this issue. In addition to concentrating on the accuracy aspect, a machine learning-based approach was also designed, reducing the processing time consumed while detecting the antipatterns.

In [16, 17], an access plan recommendation method using SQL queries was proposed to analyze the similarity between queries. Yet another method was used in [2, 18, 19] by employing a pair-wise similarity matrix and membership degree for increasing the interclass similarity and minimizing intraclass similarity. However, optimal clusters were not said to be formed.

Although reasonable work has been provided for antipattern detection, this work ensures minimum time complexity with higher antidetection accuracy with the help of ensemble methods. The next section proposes the Random Forest Bagging X-means SQL Query Clustering (RFBXSQLQC) methodology for antipattern detection.

3. Methodology

Query logs are said to be enormous in quantity and hence found to be challenging in extracting the patterns from a set of queries. Clustering is employed to obtain a thorough understanding of large query logs where the patterns for analysis are split into different unique groups for determining antipatterns. Here, mobile database is taken into account to frame the query logs. In this work, a technique called Random Forest Bagging X-means SQL Query Clustering (RFBXSQLQC) is proposed for antipattern detection. Figure 1 given below shows the block diagram of the RFBXSQLQC technique.

As illustrated in Figure 1, two different steps are involved. They are clustering using Random Forest Bagging X-means and voting using Random Forest Bagging-based Voting model. First, a different number of patterns (i.e., queries) are obtained with the input SQL query log dataset. With the obtained input patterns, several bootstrap samples are formed.

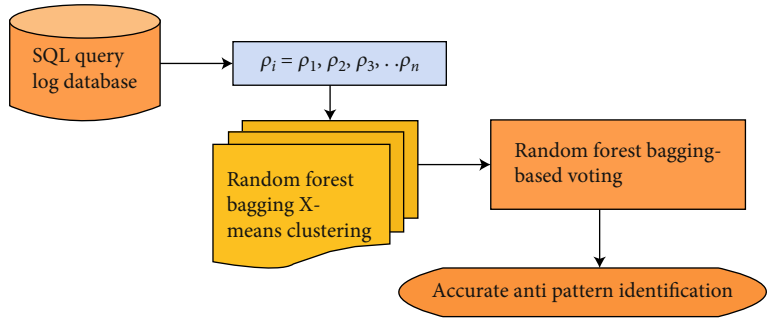

Figure 1: Block diagram of Random Forest Bagging X-means SQL Query Clustering.

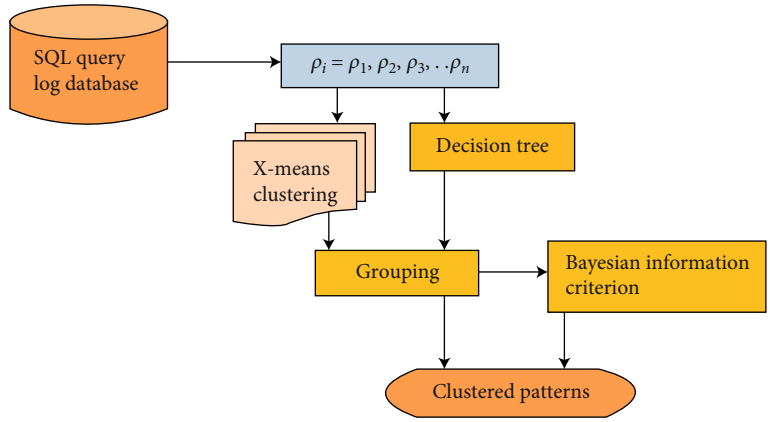

Figure 2: Block diagram of Random Forest Bagging X-means Clustering model.

Followed by which pattern clustering is performed using Random Forest Bagging X-means. Next, a voting scheme is applied to the clustered patterns and according to the results of the voting, similar patterns and antipatterns are significantly obtained.

3.1. Random Forest Bagging X-Means Clustering Model. Even though many clustering methods are used in the existing works like decision tree [20], machine learning-based ensemble methods [21], and Bayesian approach [22], in this section, Random Forest Bagging X-means Clustering model was selected as one of the best models and is presented where the clustering of patterns is performed. Different unique weak clusters are formed during this process based on the similarity between cluster weight value and input patterns. Figure 2 shows the block diagram of the Random Forest Bagging X-means Clustering model.

As shown in the above figure, let consider, "n" number of patterns or queries collected from the SQL query log dataset. This is mathematically expressed as given below.

$$\rho_i = \rho_1, \rho_2, \rho_3, ..\rho_n \in D^p. \quad (1)$$

From the above equation (1), "ρ_i" represents the set of input patterns "$\rho_1, \rho_2, \rho_3, ..\rho_n$" for the respective SQL query log dataset, "D^p" with IIT Bombay dataset provided as input. For each set of input patterns, bootstrap samples are formed. Followed by which to identify the antipatterns, X-means clustering is applied to form weak clusters. Initially, in weak X-means clustering, "x" number of clusters are framed, and this is mathematically expressed as given below.

$$C_i = c_1, c_2, c_3, ..c_x. \quad (2)$$

From the above equation (2), the total cluster set is represented as "C_i", with "x" number of clusters represented as "$c_1, c_2, c_3, ..c_x$"$c_1, c_2, c_3, \cdots .c_x$. Followed by which, for each "x" number of clusters, the value of weight is allocated in an arbitrary pattern. Then, the similarity between cluster weight value and input patterns is estimated by integrating X-means clustering and decision tree. With the training patterns provided as input, the decision tree identifies the relevant and irrelevant patterns for performing grouping. As far as decision tree is concerned, the root node is referred to as the start node of the tree, and the terminal node is referred to as the leaf node, and with the probability of a node being split into other subparts, then it is referred to as the child nodes.

In [23], detection of antipatterns was performed via a cluster-based decision tree. In contrast, the execution of pattern identification was performed by employing knowledge according to the student actions from the SQL query error database. Despite improvement in clustering accuracy, the antipattern detection accuracy was not focused. In this work, a decision tree with a dice similarity coefficient is utilized to address this issue. Here, the similarity between the cluster weight and input patterns is determined, improving antipattern detection accuracy. This is mathematically expressed as given below.

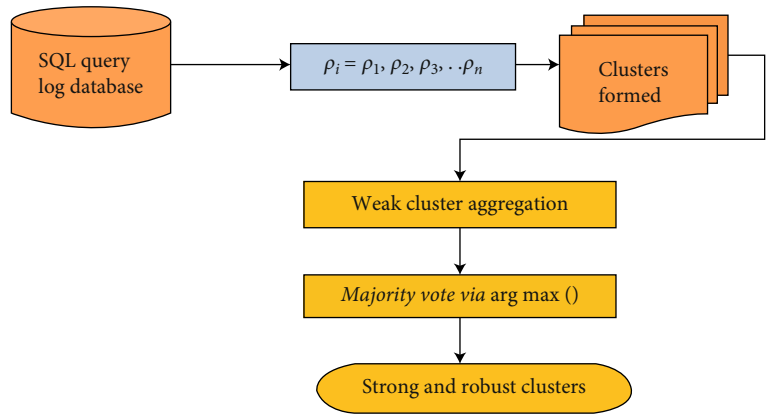

Figure 3: Block diagram of Random Forest Bagging-based Voting model.

$$D(\rho_i, c_w) = \frac{\rho_i \bigcap c_w}{|\rho_i| * |c_w|}. \tag{3}$$

From the above equation (3), at the root node for each pattern, the dice similarity coefficient "$D(\rho_i, c_w)$" is evaluated using the cluster weight "c_w," with the intersection symbol "$\bigcap$" representing a mutual dependence between input pattern "ρ_i" and cluster weight "c_w." On the other hand, "$|s_i|$" and "$|\tau_j|$" refer to the cardinalities between input pattern "ρ_i" and cluster weight "c_w," respectively. Finally, at the child node, the resultant value of dice similarity coefficient "$D(\rho_i, c_w)$" is obtained ranging between "0" and "1." Upon similarity of the weight and input pattern, then, the output of "$D(\rho_i, c_w)$" is said to be "1," otherwise, called the relevant pattern. On the other hand, if the weight and input pattern are dissimilar, then, the output of "$D(\rho_i, c_w)$" is said to be "0," otherwise, called the irrelevant pattern. Based on this similarity value, a grouping of similar patterns is said to be made. Finally, suppose the input patterns are still not grouped to any one of the clusters. In that case, the Bayesian information criterion is applied to improve the clustering performance via repeatedly conducting subdivisions. In this manner, the entire input patterns are grouped. The Bayesian information criterion is mathematically expressed as given below.

$$X_{\text{Means}}\text{Cluster} = \arg \sum_{i=1}^{X} \sum_{\rho_i \in c_i}^{x} D(\rho_i, c_w). \tag{4}$$

From the above equation (4), the input pattern set belonging to cluster "i" is denoted by "c_i." With the aid of the above equation, input patterns are allocated to unique clusters in such a manner whose similarity from cluster weight is greater than all cluster weights. In this manner, all the given input patterns are grouped into either relevant or irrelevant clusters. The process is an aid to be repeated until the clustering of all the input patterns is performed. Though the subdivisions were made as mentioned in [24–27], weak cluster results' accuracy was insufficient in detecting the antipatterns. A bagging ensemble algorithm improves the weak cluster accuracy results with a minimum false positive rate.

3.2. Random Forest Bagging-Based Voting Model. Several machine learning techniques were applied by different research academicians as given in [1, 28, 29] to predict antipatterns from databases. However, the processing time involved in antipattern detection was found to be high. Hence, an ensemble model is performed to minimise the time complexity involved in the process and obtain strong clustering results. Figure 3 shows the block diagram of the Random Forest Bagging-based Voting model.

As shown in the above figure, let us consider X-means clustering constructs "x" number of weak clusters with results obtained for each pattern in bootstrap samples. Then, all the weak clusters are aggregated into a strong cluster by using the following mathematical representation.

$$B(\rho_i) = B_1(\rho_i) + B_2(\rho_i) + \cdots + B_x(\rho_i). \tag{5}$$

From the above equation (5), the aggregated results of weak clusters "$B_1(\rho_i) + B_2(\rho_i) + \cdots + B_x(\rho_i)$" are stored in the resultant "$B(\rho_i)$" for further processing. For the aggregated weak clusters, a vote "v_i" is applied, and this is mathematically expressed as given below.

$$v_i \longrightarrow \sum_{i=1}^{x} B(\rho_i). \tag{6}$$

Finally, the majority vote is combined with weak X-means clustering results to produce a strong clustering outcome for accurately grouping query patterns with a minimum error rate. The strong clustering output is mathematically expressed as given below.

$$X(\rho_i) = v(B(\rho_i)). \tag{7}$$

From the above equation (7), the strong clustering results are obtained and stored in "$X(\rho_i)$" based on the majority votes of the weak cluster formed using the arg max function "v'." With the strong clustering results, the antipatterns are said to be extensively obtained with a greater amount of accuracy and minimum time complexity. The pseudocode representation of Random Forest Bagging X-means Clustering is given below.

Input: Number of patterns (i.e. queries).'ρ_i, $i = 1, 2, 3..n$'.
Output: Accurate and timely anti-pattern identification.
1: **Initialize** 'x' number of weak X-means clusters'C_i'.
2: **Begin.**
3: **For** each query pattern'ρ_i'.
4: Randomly assign weight to cluster 'C_i'.
5: Measure similarity between cluster weight 'c_w' and input patterns 'ρ_i'.
6: **If** similarity coefficient '$D(\rho_i, c_w)$' is 1, then 'c_w' and 'ρ_i' are more similar.
7: Else 'c_w' and 'ρ_i' are dissimilar.
8: **End if.**
9: Group similar patterns 'ρ_i' into cluster 'C_i'.
10: **If** 'ρ_i' not grouped into cluster apply Bayesian information criterion to group patterns 11: **Else.**
12: Combine all the weak clusters results using (5).
13: Apply voting using (6).
14: **Return** (anti-patterns).
15: **End if.**
16: **End for.**
17: **End.**

Algorithm 1: Random Forest Bagging X-means Clustering.

Table 1: Tabulation of antipattern detection accuracy.

Number of queries	Antipattern detection accuracy (%)			
	Existing cluster-based decision tree approach	Existing pattern detection method	Existing GPT	Proposed RFBXSQLQC technique
50	86	84	82	96
100	82	81	80	95
150	81	78	76	93
200	78	76	74	91
250	77	75	73	89
300	80	78	77	92
350	83	81	80	95
400	85	83	82	96
450	82	80	79	92
500	86	83	82	95

As given in the above Random Forest Bagging X-means Clustering algorithm, the objective here remains in identifying the antipatterns with minimum time and maximum accuracy. Initially, the bootstrap samples are created for each pattern with the different unique number of patterns obtained from the query log dataset. Then, a weak clustering called X-means clustering is utilized to group the patterns into the relevant and irrelevant sets. In addition, the resultant values of all the weak clusters are integrated to generate a strong clustering outcome. Moreover, for each resultant weak cluster, a voting scheme is utilized, and from that cluster, the cluster with a majority vote is selected for obtaining accurate clustering results. This, in turn, aids in grouping the patterns into relevant and irrelevant.

4. Experimental Settings

In this section, the experimental settings of the proposed Random Forest Bagging X-means SQL Query Clustering (RFBXSQLQC) technique are performed using Java language. To conduct a fair comparison, three existing methods, namely, cluster-based decision tree technique [1], pattern detection method [2], and graph-based database partitioning method called GPT [3], are taken into account along with the proposed RFBXSQLQC technique since similar numbers of queries are utilized with a simulation of 10 runs. The performance of the proposed RFBXSQLQC technique is analyzed with the aid of the IIT Bombay dataset. A total of 630 instances are presented in the IIT Bombay dataset that comprises answers to a set of SQL questions obtained from the student in the undergraduate databases course. It includes three columns like ID, label, and query. SQL queries are collected from the IIT Bombay dataset, and accordingly, experiments are conducted by considering the metrics such as

(i) Antipattern detection accuracy

(ii) Time complexity

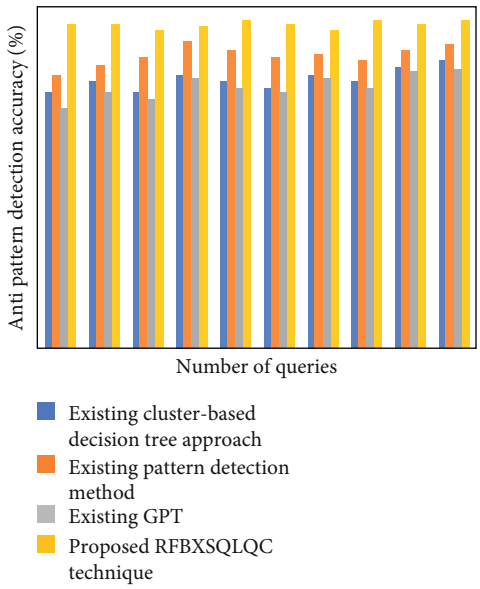

Figure 4: Graphical representation of antipattern detection accuracy.

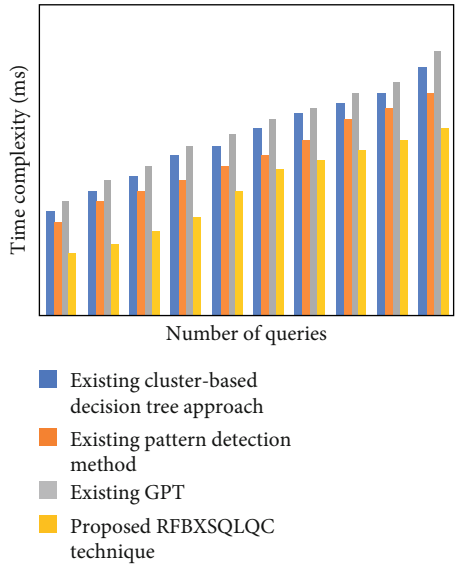

Figure 5: Graphical representation of time complexity.

(iii) False-positive rate

(iv) Computational overhead

To improve the accuracy of query log data with respect to different number of queries.

5. Results and Discussion

5.1. Performance Analysis of Antipattern Detection Accuracy. Antipattern detection accuracy is the percentage ratio of a number of queries properly grouped as relevant or irrelevant to the total number of queries obtained as input for simulation. This is mathematically expressed as given below.

$$\text{APDA} = \frac{N_{AD}}{N} * 100. \tag{8}$$

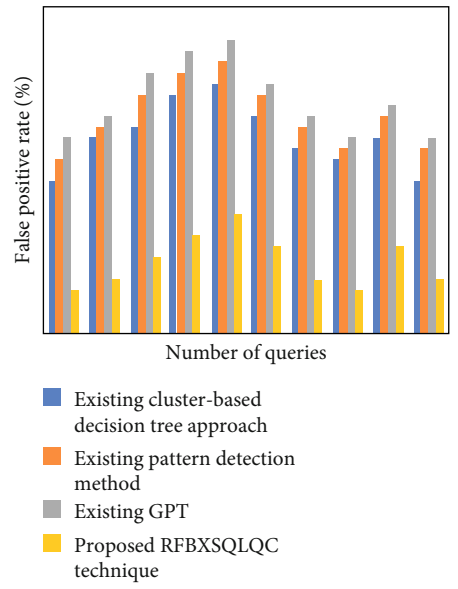

Figure 6: Graphical representation of the false-positive rate.

From the above equation (8), the antipattern detection accuracy rate "*APDA*" is obtained based on the number of queries accurately detected. "N_{AD}" to the sample input queries "*N*." It is measured in terms of percentage (%). Table 1 given below shows the antipattern detection accuracy using RFBXSQLQC technique, cluster-based decision tree technique [1], pattern detection method [2], and graph-based database partitioning method called GPT [3].

Figure 4 above shows the graphical representation of antipattern detection accuracy for different queries obtained from the IIT Bombay dataset. Here, the *x*-axis provides the number of queries, and the *y*-axis gives the antipattern detection accuracy for the proposed and existing three methods. The graphical representation shows that higher antipattern detection accuracy is observed in the proposed RFBXSQLQC technique than in the existing methods. The better accuracy in the RFBXSQLQC technique is due to the consideration of clustering of input patterns via X-means clustering. Here, the patterns are grouped into two sets by estimating the similarity between cluster weight and input patterns. This is followed by which bagging ensemble algorithm is applied to combine all the clustering results to obtain the stronger output. With this, the antipattern detection accuracy is increased in the proposed RFBXSQLQC technique using IIT Bombay dataset by 14%, 17%, and 19% compared to the existing cluster-based decision tree approach existing pattern detection method and existing GPT, respectively.

5.2. Performance Analysis of Time Complexity. Time complexity refers to the amount of time consumed in identifying the antipatterns via the clustering process and is mathematically expressed as given below.

$$\text{Time complexity} = N * \text{Time}(\text{CSQ}). \tag{9}$$

From the above equation (9), "*N*" represents the input

Table 2: Tabulation of space complexity.

Number of queries	Space complexity (MB) Existing cluster-based decision tree approach	Existing pattern detection method	Existing GPT	Proposed RFBXSQLQC technique
50	27	24	29	16
100	37	31	38	23
150	46	39	48	30
200	58	48	60	36
250	67	59	69	44
300	74	67	76	55
350	81	77	83	63
400	92	85	94	72
450	98	93	100	80
500	106	101	108	90

queries, and "Time(CSQ)" represents the clustering of the query. It is expressed in terms of milliseconds (ms).

Figure 5 given above shows the graphical representation of time complexity with respect to the differing numbers of queries in the range of 50 to 500 obtained at different time intervals. From the above figure, it is inferred that the time complexity is significantly reduced using the proposed RFBXSQLQC technique when compared to other existing methods. This is because of X-means clustering applied in the RFBXSQLQC technique that groups the patterns based on the dice similarity coefficient. Here, the similarity between the weights of each cluster for each pattern and the actual numbers of patterns are utilized in grouping the patterns as referred to in [30]. With this, the proposed RFBXSQLQC technique's time complexity is minimized for the IIT Bombay dataset by 30%, 22%, and 34%, respectively, compared to existing methods.

5.3. Performance Analysis of the False-Positive Rate. The false-positive rate is the percentage ratio of the number of queries incorrectly clustered to the total sample input queries provided as input. This is mathematically expressed as given below.

$$\text{FPR} = \frac{N_{\text{incorrectly clustered}}}{N} * 100. \tag{10}$$

From the above equation (10), the false-positive rate "FPR" is measured based on the total input queries "N" and the number of incorrectly clustered queries "$N_{\text{incorrectly clustered}}$." It is represented in terms of percentage (%).

Figure 6 given above shows the graphical representation of the false-positive rate for different numbers of queries. From the figure, it is inferred that the false-positive rate is gradually reduced using all four methods. However, comparative analysis shows better using the proposed RFBXSQLQC technique compared to the other three existing methods. The minimal false-positive rate is arrived at in the proposed RFBXSQLQC technique by employing the decision tree in the X-means clustering process. Here, with the aid of the decision tree, significant decision regarding the relevancy or irrelevance of the query is made via the similarity measure, and more similar patterns were found to be grouped using the RFBXSQLQC technique. Due to this, the false-positive rate of the RFBXSQLQC technique using the IIT Bombay dataset was said to be reduced by 64%, 68%, and 70%, respectively, than the existing methods.

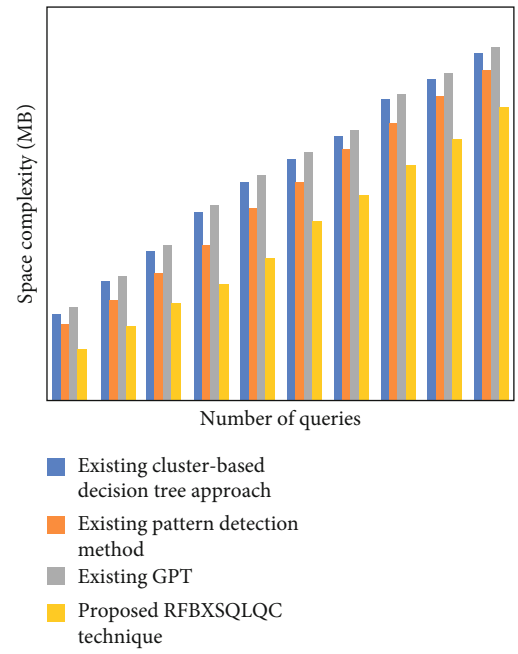

Figure 7: Graphical representation of space complexity.

5.4. Performance Analysis of Computational Overhead. Computational overhead refers to the memory consumed in storing similar queries for providing clustering outcomes, which is mathematically expressed as given below.

$$SC = N * \text{memory (storing single query)}, \tag{11}$$

From the above equation (11), the space complexity "SC"

is measured based on the number of queries "*N*" and the memory consumed for the corresponding queries. It is measured in terms of megabytes (MB). Table 2 given below shows the space complexity using RFBXSQLQC technique, cluster-based decision tree technique [1], pattern detection method [2], and graph-based database partitioning method called GPT [3].

Figure 7 given above illustrates the graphical representation of space complexity for differing numbers of queries in the range of 50 to 500 obtained at different time intervals. From the figure, it is inferred that the RFBXSQLQC technique achieves better results in the reduction of space complexity when compared to the other three existing methods. The improvement in the RFBXSQLQC technique is due to the utilization of random forest bagging X-means clustering process for clustering the given input patterns. With the assistance of this clustering process, higher rates of similar patterns are grouped accurately. With this, the space complexity using the proposed RFBXSQLQC technique for the IIT Bombay dataset is reduced by 29%, 21%, and 31%, respectively, than the existing methods.

6. Conclusion

In this work, an antipattern detection technique with a high accuracy rate and less time complexity is designed using the RFBXSQLQC technique. The contribution of this technique remains in proposing a random forest bagging X-means clustering with the objective of grouping similar patterns for obtaining the antipatterns. With the application of the Random Forest Bagging X-means Clustering model, the percentage of a number of patterns was said to be accurately grouped than the state-of-the-art works. Also, by utilizing X-means clustering, the overall time complexity involved in the pattern clustering process was found to be lesser. Moreover, to minimise the computational overhead incurred during the clustering, the Random Forest Bagging-based Voting model was applied that minimizes the number of patterns incorrectly grouped. Experimental analysis of the RFBXSQLQC technique was also carried out with the metrics such as antipattern detection accuracy, time complexity, false-positive rate, and computational overhead with respect to the differing number of queries that showed improvement when compared to the state-of-the-art works.

Even though the proposed work outperforms existing works in retrieving the mobile data through query logs, a very limited number of query logs are considered for the experiment. In the future, the same should experiment with a large data set and complex query logs, which may solve real-world problems.

Data Availability

Public domain data is used for the work. Data can be found on http://www.cse.iitb.ac.in/infolab/xdata/

Conflicts of Interest

There is no conflict of interest for authors.

Acknowledgments

Symbiosis International University, India funded the research work.

References

[1] A. Lino, A. Rocha, L. Macedo, and A. Sizo, "Application of clustering-based decision tree approach in SQL query error database," *Future Generation Computer Systems*, vol. 93, pp. 392–406, 2019.

[2] N. Arzamasova, M. Schaler, and K. Bohm, "Cleaning antipatterns in an SQL query log," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 3, pp. 421–434, 2018.

[3] Y. Nam, D. Han, and M. Kim, "A parallel query processing system based on graph-based database partitioning," *Information Sciences*, vol. 480, pp. 237–260, 2019.

[4] I. Couso and L. Sánchez, "A note on "similarity and dissimilarity measures between fuzzy sets: a formal relational study" and "additive similarity and dissimilarity measures"," *Fuzzy Sets and Systems*, vol. 390, pp. 183–187, 2020.

[5] A. Ouni, M. Kessentini, K. Inoue, and M. Cinneide, "Search-based web service antipatterns detection," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 603–617, 2015.

[6] A. Barbez, F. Khomh, and Y. Guéhéneuc, "A machine-learning based ensemble method for anti-patterns detection," *Journal of Systems and Software*, vol. 161, article 110486, 2020.

[7] Y. Fan and X. Lu, "An online Bayesian approach to change-point detection for categorical data," *Knowledge-Based Systems*, vol. 196, article 105792, 2020.

[8] Z. Kermansaravi, M. Rahman, F. Khomh, F. Jaafar, and Y. Guéhéneuc, "Investigating design anti-pattern and design pattern mutations and their change- and fault-proneness," *Empirical Software Engineering*, vol. 26, no. 1, 2021.

[9] B. Chandra, M. Joseph, B. Radhakrishnan, S. Acharya, and S. Sudarshan, "Partial marking for automated grading of SQL queries," *Proceedings of the VLDB Endowment*, vol. 9, no. 13, pp. 1541–1544, 2016.

[10] B. Chandra and S. Sudarshan, "Runtime optimization of join location in parallel data management systems," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1490–1501, 2017.

[11] C. Trubiani, A. Bran, A. van Hoorn, A. Avritzer, and H. Knoche, "Exploiting load testing and profiling for performance antipattern detection," *Information and Software Technology*, vol. 95, pp. 329–345, 2018.

[12] F. Sabir, G. Rasool, and M. Yousaf, "A lightweight approach for specification and detection of SOAP anti-patterns," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 5, 2017.

[13] R. Kumar Dhanaraj, L. Krishnasamy, O. Geman, and D. Roxana Izdrui, "Black hole and sink hole attack detection in wireless body area networks," *Computers, Materials & Continua*, vol. 68, no. 2, pp. 1949–1965, 2021.

[14] L. Liang, "Design and implementation of a cross-database query tool on multi-source databases," *Journal of Computers*, vol. 13, no. 9, pp. 1042–1052, 2018.

[15] J. Zheng and X. Shen, "Pattern mining and detection of malicious SQL queries on anonymization mechanism," *IEEE Access*, vol. 9, pp. 15015–15027, 2021.

[16] K. Kuru and W. Khan, "Novel hybrid object-based non-parametric clustering approach for grouping similar objects

in specific visual domains," *Applied Soft Computing*, vol. 62, pp. 667–701, 2018.

[17] M. Hirsch, A. Rodriguez, J. Rodriguez, C. Mateos, and A. Zunino, "Spotting and removing WSDL anti-pattern root causes in code-first web services using NLP techniques: a thorough validation of impact on service discoverability," *Computer Standards & Interfaces*, vol. 56, pp. 116–133, 2018.

[18] S. Saluja and U. Batra, "Assessing quality by anti-pattern detection in web services," in *Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM)*, Amity University Rajasthan, Jaipur - India, 2019.

[19] M. D. Ramasamy, K. Periasamy, L. Krishnasamy, R. K. Dhanaraj, S. Kadry, and Y. Nam, "Multi-disease classification model using Strassen's half of threshold (SHoT) training algorithm in healthcare sector," *IEEE Access*, vol. 9, pp. 112624–112636, 2021.

[20] A. Akila and M. R. Padma, "Breast cancer tumor categorization using logistic regression, decision tree and random forest classification techniques," *International Journal of Research in Arts and Science*, vol. 5, pp. 282–289, 2019.

[21] L. Chen, Y. Lin, J. Wang, H. Huang, D. Chen, and Y. Wu, "Query grouping-based multi-query optimization framework for interactive SQL query engines on Hadoop," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 19, article e4676, 2018.

[22] L. Krishnasamy, R. Dhanaraj, D. Ganesh Gopal, T. Reddy Gadekallu, M. Aboudaif, and E. Abouel Nasr, "A heuristic angular clustering framework for secured statistical data aggregation in sensor networks," *Sensors*, vol. 20, no. 17, p. 4937, 2020.

[23] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," *Journal of Systems Architecture*, vol. 112, article 101861, 2021.

[24] A. Lino, Á. Rocha, and A. Sizo, "Virtual teaching and learning environments: automatic evaluation with artificial neural networks," *Cluster Computing*, vol. 22, no. S3, pp. 7217–7227, 2019.

[25] D. Farid, L. Zhang, C. Rahman, M. Hossain, and R. Strachan, "Hybrid decision tree and naive Bayes classifiers for multiclass classification tasks," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1937–1946, 2014.

[26] A. Kicsi, V. Csuvik, and L. Vidacs, "Large scale evaluation of natural language processing based test-to-code traceability approaches," *IEEE Access*, vol. 9, pp. 79089–79104, 2021.

[27] P. Chinnasamy, S. K. Udgata, K. Lalitha, and A. Jeevanantham, "Multi-objective based deployment of throwboxes in delay tolerant networks for the internet of things environment," *Evolutionary Intelligence*, vol. 14, no. 2, pp. 895–907, 2021.

[28] D. Pradeep and C. Sundar, "QAOC: novel query analysis and ontology-based clustering for data management in Hadoop," *Future Generation Computer Systems*, vol. 108, pp. 849–860, 2020.

[29] R. Jiang, R. Lu, and K. Choo, "Achieving high performance and privacy-preserving query over encrypted multidimensional big metering data," *Future Generation Computer Systems*, vol. 78, pp. 392–401, 2018.

[30] M. Poongodi and S. Bose, "Design of intrusion detection and prevention system (IDPS) using DGSOTFC in collaborative protection networks," in *2013 Fifth International Conference on Advanced Computing (ICoAC)*, pp. 172–178, Chennai, India, 2013.