[1] Dr. Eugene S. Valeriano

Deduplication Methods Using Levenshtein Distance Algorithm

Abstract: - The study aimed to propose methods to improve the data integrity of the Relational databases such as MS SQL, MySQL and PostgreSQL via record duplication detection. The FODORS and ZAGAT Restaurant database benchmark datasets have been utilized to facilitate the processes involved in preparing and delivering high-quality data. Furthermore, the Levenshtein distance algorithm was used to propose three (3) methods namely: default, eliminating equal string, and knowledge-based libraries to cut duplicate records in the database. In the 70% selected threshold, the average detected duplicate records of 88 out of 112 records in the restaurant dataset. Finally, to efficiently detect duplicate records in the database, depend on the data being analyzed and threshold selected.

Keywords: data, efficiency, information, quality, record.

I. INTRODUCTION

These days, information is stored on computer systems, from which data can be readily moved and altered. Considering saved data from these records is occasionally redundant and consumes storage space, the methodical use of data storage and search is essential. Retrieving records from various data structures—huge databases that are being implemented—that need search operations in order to carry out CRUD functions is a key software application need [1].

Advocated the use of TempTable and Identity Column as strategies to eliminate duplicate records. When putting records into a table in the relational database management system SQL Server, it has been determined that there are some duplicate records. Table primary key or cluster-index has not yet created which is the cause. In this instance, there is a breach in the data integrity, which could lead to inaccurate results, duplicate record entry, and wasteful data outcomes. This results in a significant quantity of storage space being consumed. Thus, determining the storage space, which is lesser than the two methods, is essential. It is also necessary to create a solution to SQL Server for managing duplicate records (Yue, 2018).

Furthermore, user input, discrepancies, null values, and incorrect spelling into a limited set of information could all have an impact on data efficiency [3]. These issues hinder the gathering of precise analysis data and the conduct of research that yields false conclusions. It was claimed that the most essential step in the record information analysis procedure to solve such a challenging challenge is data processing. Moreover, the suggested techniques fall into four groups: (i) data cleansing, which attempts to complete missing values, (ii) data integration (iii) aggregation methods and normalization that include change of data, and (iv) data removal that goes to minimize the amount of data.

Meanwhile, Levenshtein distance is used when evaluating similar string that requires a knowledge base which was mentioned by [4], wherein Road can be abbreviated to Rd. and some other instance like 7th for Seventh, which Levenshtein distance cannot determine.

Levenshtein edit distance, which employs deletion, insertion, and substitution to calculate the number of edit distances, is maybe a string metric that works well for comparing text texts. The source string must be changed to the target string in this way. This algorithm can find the similarity of documents to detect suspected research manuscripts on the internet [5]. On the other hand, the Damerau-Levenshtein distance is also used in determining edit distance between two strings. The only edge is that it has another property of transposition of adjacent character and is used in string correction problems [6].

On the other hand, profiling the query result to the local system can also assist reduce the number of queries that the database must execute and the amount of network, database, and server resources used. This will speed up user requests and prevent redundant data from being stored in the repository [7].

[1] Tarlac Agricultural University, Santa Ignacia, Philippines

* Corresponding Author Email: esvaleriano@tau.edu.ph

With this, the Philippine Government can significantly receive help from the proposed study. Government agencies do not have a centralized data system that causes duplication of citizen's information. But with the help of the proposed method, duplication of data in the database can be imposed and verified efficiently.

The Philippine government pushes the National ID System for all the citizens to have a unique ID to help prevent committing mistakes in tracing persons who are suspected with COVID-19. According to Sen. Sherwin Gatchalian, the National ID, often, is prone to duplication and prone to fraud. Still, the local and national government enable this to avoid all this fraud and duplication. Besides, the Philippine Statistics Authority (PSA) and in the fourth quarter of 2019 the government started testing the National ID System with an agreement to finish population enrollment by the middle of 2022 [8]. The National ID also helps to streamline COVID-19 to speed up the release of aid recipients in the problem of Social Amelioration Funds (SAF) distribution to solve any discrepancies between Local Government Units (LGUs) and the Department of Social Welfare Development (DSWD) database [9].

Also, universities can benefit from the proposed study since the retrieval of data of students, teaching, and non-teaching staff can be time-consuming, especially when dealing with a large volume of information. The integrity of students' data can also be applied using the method. The operational cost can be minimized using the innovative approach.

Libraries can also benefit from the study. Retrieving books can be done quickly and effectively. The process of collecting, sorting, and finding books and other information can be done promptly and effectively in just a brief period.

Likewise, the findings of the study may be a prodigious help to Database Administrators since a new maintenance mechanism will be introduced. Working on databases will be implemented to save memory usage and storage space. This suggested approach encourages the researcher to enhance the SQL database's data integrity through record duplicate identification and deletion [2]. The Structured Query Language performs better for particular statements and joint tables. Furthermore, the database's record integrity will be preserved, along with the reduction of storage space and RAM allocation to the server.

II. Experimental and Computational Details

A. *Methodology*

The Restaurant Dataset (FODORS and ZAGAT) [10] are a collection of benchmark datasets for evaluating methods and algorithms. Each dataset is included of .csv files and a defined number of duplicate records. From this dataset, the data in the MS SQL Server Database were imported for usage and analysis in the experiments conducted.

The study implemented and used the Levenshtein distance algorithm for detecting records in the database. The researcher formulated and implemented three (3) proposed methods for the experiment in finding duplicate records in the database. The first one is Default which is the natural implementation of the Levenshtein distance; (2) EES stands for Elimination of Equal String method that eliminates equal string found in the comparison; (3) KBL – is also based on the Levenshtein distance, but the method is improved in finding similar terminologies via Knowledge-Based Libraries [4]. The KBL method adopted the Rule-based technique of [11]. On the other hand, the three methods implemented the Blocking and Windowing technique to improve the efficiency of comparison and to speed up the data comparison process [12].

B. *Study Design*

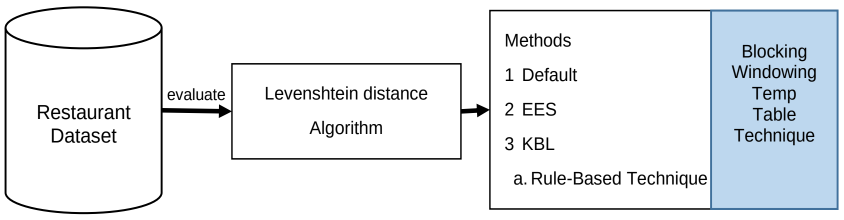


Fig. 1: Conceptual Framework.

Fig. 1, illustrates the propose method that can be the improvement of the data Integrity of SQL Database via Record Duplication Detection. As shown in the diagram the Restaurant, Product, and Bibliography benchmark datasets were used to evaluate the data to detect duplicate record in the database along with the proposed three methods namely: default, eliminating equal string (EES) and Knowledge-based libraries (KBL). This was done with the use of adopted Rule-Based Technique to improve Levenshtein Algorithm to find similar terminology of string. The Blocking, Windowing were used to manage the performance of the comparison.

C. *Datasets*

Restaurant Dataset from ZAGAT and FODOR'S firm offered a benchmark dataset for evaluating the methodologies within a relational database of Levenshtein distance function.[10]

Their algorithms and techniques for removing data redundancy in relational databases were evaluated using a variety of research studies [10], [13] , [4], [14]

The dataset supplier states that the documents were marked with the following column fields of the database: id, name, address, phone, city, and type and shown 112 duplicates out of a total of 864 records. Table I contains a list of the data that was used to assess the techniques for finding duplicate records in SQL databases.

Table I. List of Tables used in the experiment for record duplication detection.

Dataset	Size	Duplicates
Restaurant	864	112

D. *Record Duplication Detection*

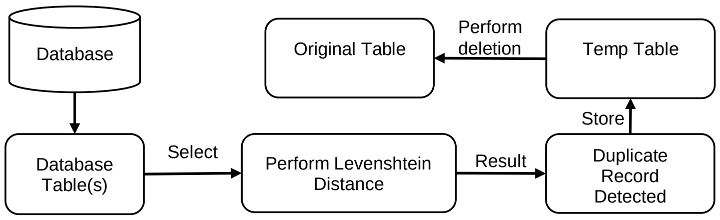


Fig. 2: The Diagram for record duplication detection and deletion.

Fig. 2, shows the diagram for record duplication detection and deletion in the database using Levenshtein distance. The researcher selected a database and then got all the tables inside the database to analyze and perform Levenshtein distance function to detect similar tuples in the table selected. The detected similar records based on a given threshold were considered as duplicate records in a table and were then stored to a temporary table, which follows the method of storing duplicate data from [2].

E. *Executing the Levenshtein distance Algorithm*

Definition of Parameters of the proposed method in Executing Levenshtein distance

\$dbName	variable that holds the database name
\$numberofTables	variable that holds the number of tables inside the database
\$tableName	holds the table name information
\$recordCount	retains the quantity of entries in the table.
\$counter	variable that increases via the table's numerical entries.
\$x	variable used to increment through the number of tables in the database
\$threshold	variable that holds to determine the percentage of the duplication detection of the Levenshtein distance algorithm
\$distance	variable that holds the distance of the comparison
\$percent	holds the percentage value of the calculation result of the similarity of the two string
\$max	retains the character count that separates the source and target strings.

The pseudocode below explains the proposed method in executing the Levenshtein distance algorithm to detect duplicate record in the database.

```
Start the Program
Set variable $recordCount = 0
Set variable $counter = 0
Set variable $x = 0
Initialize variable $dbName
Initialize variable $numberofTables
Initialize variable $threshold
Get list of tables
Loop through each table in the database
Set the variable $tableName to the name of the table
Set the variable $recordCount to the total number of record in the table
Loop through each record in the table
Set the variable $counter is equal to zero
Get the distance between the source string and target string
Set the variable $percent to the result of the percentage of the similarity
If percent is greater than or equal to the threshold
Save to Temporary Table
If counter is equal to the total record count of the table
If variable $x is equal to the total number of tables count
End the Program
Else
Add 1 to the variable $x
Input the next Table
End If
Else
Add 1 to the variable $counter
Input the next Record
End If
Else
If counter is equal to the total record count of the table
If variable $x is equal to the total number of tables count
End the Program
Else
Add 1 to the variable $x
Input the next Table
End If
Else
Add 1 to the variable $counter
Input the next Record
End If
End If
End Loop
End Loop
```

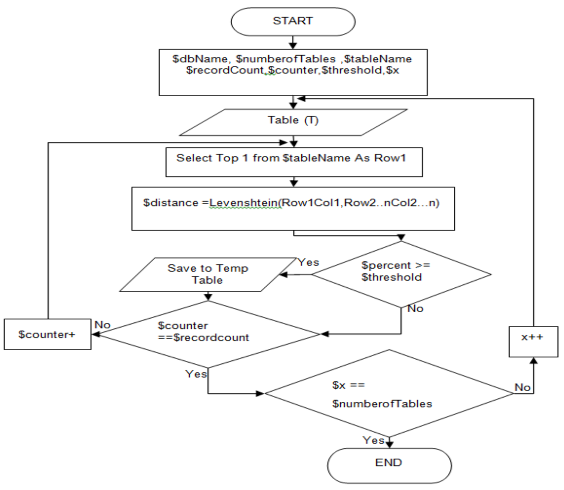

START
$dbName, $numberofTables ,$tableName $recordCount,$counter,$threshold,$x
Table (T)
Select Top 1 from $tableName As Row1
$distance =Levenshtein(Row1Col1,Row2..nCol2...n)
$percent >= $threshold
Yes
Save to Temp Table
No
$counter ==$recordcount
No
$counter+
Yes
$x == $numberofTables
No
x++
Yes
END

Fig. 3: Levenshtein distance function flowchart used to find identical records in a relational database.

Fig. 3, demonstrates the flow chart for using the SQL database's Levenshtein distance function to find duplicate records. Once the method started, it will initialize what the database needs to be analyzed, which represents the variable $dbName. The variable $numberofTables indicates how many tables are present in the database that has been chosen. The variable $x represents the counter of each loop in the number of tables. Conversely, the T stands for the chosen table.

The method performs SQL command 'Select Top 1 from the $tableName' as Row1. The first record in the table executes the Levenshtein function then calculates the distance of the succeeding records that represents the $distance variable. The columns of each tuple in the tables get the percentage of the similarity of the comparison. Once the variable $percentage were calculated, they were compared to the threshold set by the user. If the $percentage is greater than or equal than the threshold, the record is stored to the temporary table, which is considered as a duplicate to the selected record comparison.

After the data is stored in the temporary table, the researcher evaluated the variable $counter which is equal to the $recordcount variable. The $recordcount variable keeps track of how many records are in the table. If the conditions evaluated is No, the researcher continued comparing the selected record to the other record in the table. If the condition evaluated is Yes, the researcher evaluated the variable $x if the variable $numberofTables in the database is equal to the variable $x. Once the $numberofTables is equal to the variable $x, it ends the execution of the Levenshtein distance calculation. If they are not yet equal, it will continue to go to the next table for analysis to detect duplicate records in the database.

F. Increase data-comparison Speed

The researcher adopted blocking and windowing techniques in speeding the comparison of data to detect duplicated records in the database along with its tables as suggested by [15]. However, Windowing Technique was also adopted since Blocking Technique has possible downsides that could also increase the frequency of false mismatches because of the behavior of bringing together in one instance and comparing.

The Windowing (Sorted Neighborhood) Technique was developed by [16]. In order to speed up the comparison, the Sorted Neighborhood technique's output must be combined. One sliding window approach is used, and each tiny group's key must be changed [17] and [15].

G. Blocking (Conventional blocking technique) (Draisbach, n.d.)

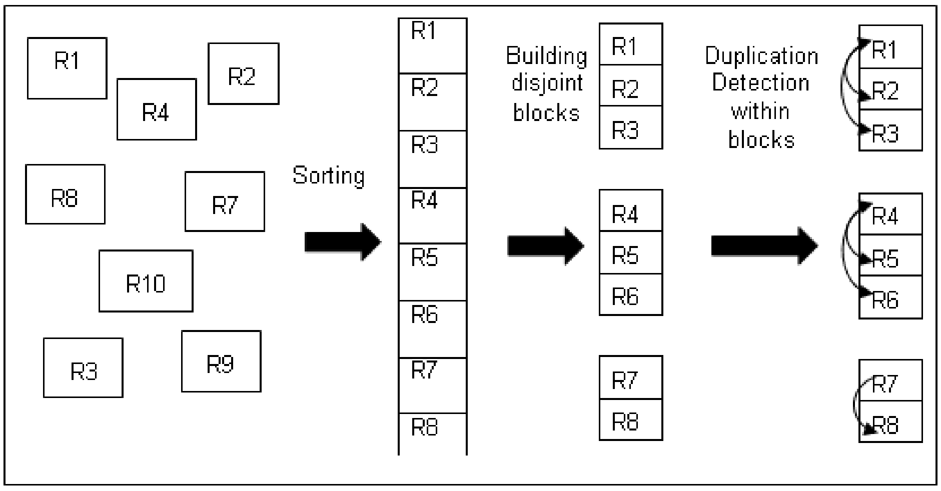


Fig. 4: Diagram for the Conventional blocking technique to increase data-comparison speed for record duplication detection

Fig. 4, shows the diagram for the implementation of the conventional blocking technique to speed-up data comparison speed between two tuples in the database to detect duplicate data. In the diagram, the first elements show that the records are not sorted. The method composes of the following three steps: (1) to order the records, (2) to build a disjoint block, and (3) to detect duplication within a single block. In this case, the process of detecting the duplicate records in the database was optimized by searching each block.

Furthermore, the method could increase the process, but it could also increase the chance of comparing other tuples' mismatch. The drawback of this method is shown in Figure 6. The method only provides faster data-comparison by its behavior of partitioning large datasets into a series of blocks. Searching of duplicate records into smaller dataset is faster rather than searching the whole dataset.

H. Windowing (Sorted Neighborhood) [18]

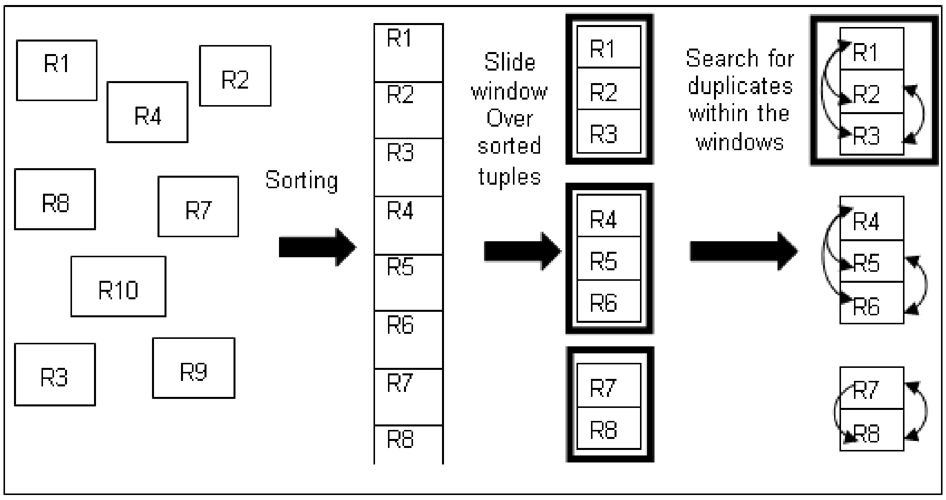

R1
R2
R4
R8
R7
R10
R3
R9
Sorting
R1
R2
R3
R4
R5
R6
R7
R8
Slide window Over sorted tuples
Search for duplicates within the windows

Fig. 5: Diagram illustrating the windowing procedure used to increase the database's record duplication detection efficiency.

Fig. 5, windowing the required collection of tuples in order, demonstrates the effectiveness of the record duplicate detection procedure. In the first group of elements, the elements are not in the same order with the blocking method. The difference of this method is once the records are sorted with the desired key, the window over sorted tuples are grouped by the fixed number of partition and are searched for duplicate records within the partition.

I. Knowledge-Based Libraries to Improve Levenshtein Distance Function

Recommends that to be able to detect duplicate records knowledge base is required [4]. Review in dealing with duplicate records [15]. One of the methods reviewed is the Rule-Based Technique [11]. On the other hand, To decide whether or not the two records match, Madnick applied the predefined heuristic rules. This technique gives one or zero results for each attribute, which is much related to the Levenshtein distance. When evaluating two-record distance and the result is zero, it means they are same. An instance of [11] defines a rule below.
IF the age is under 22 then
status as a student
Graduate status for ELSE
student_major = MIS IF course_id = 564 or course_id = 579

J. Rule-Based Technique

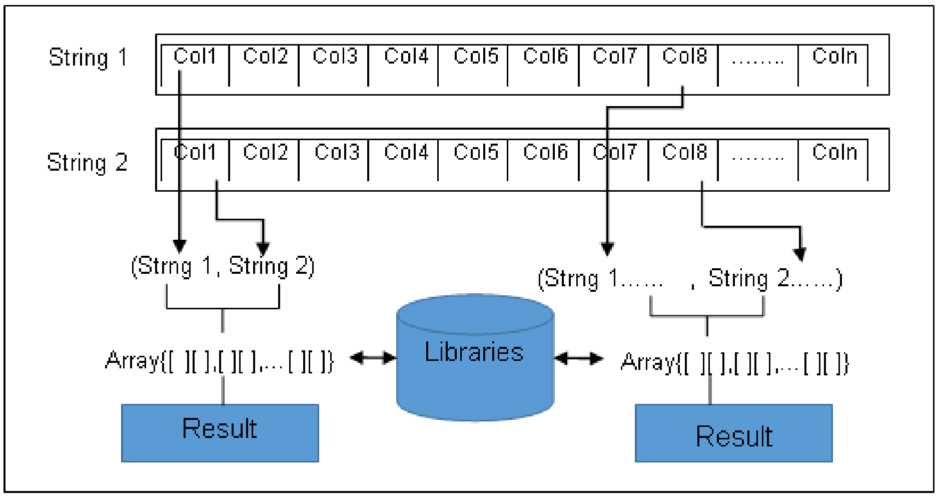
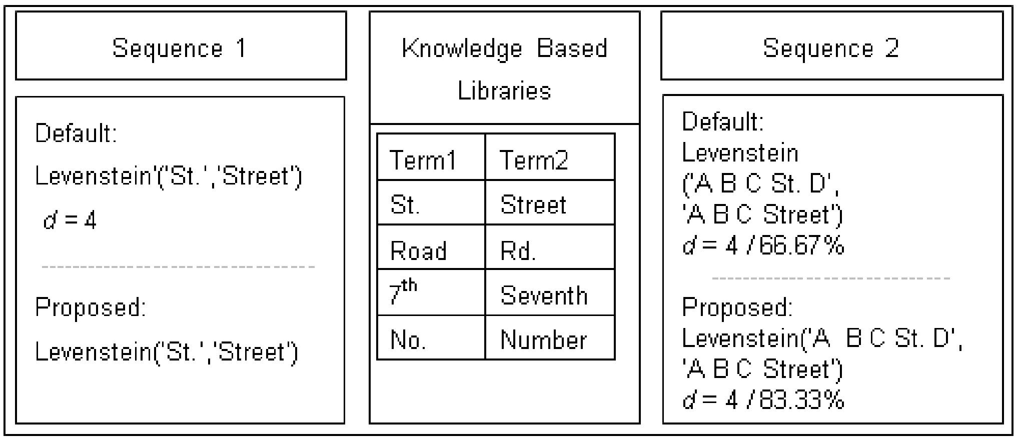

String 1
Col1 Col2 Col3 Col4 Col5 Col6 Col7 Col8 .... Coln
String 2
Col1 Col2 Col3 Col4 Col5 Col6 Col7 Col8 .... Coln
(Strng 1, String 2)
(Strng 1...... , String 2......)
Array{[ ][ ],[ ][ ],...[ ][ ]}
Libraries
Array{[ ][ ],[ ][ ],...[ ][ ]}
Result
Result

Fig. 6: Architectural design for the improvement of Levenshtein distance to detect duplicate records through knowledge based

Fig. 6, shows the architectural design is derived from the proposed method of [11] that used the heuristics rules to determine if two strings are matched. The design shows the two strings as a Tuple in the database to compare each of the columns. If the column value contains a single word and if they match within the two-dimensional array that was retrieved in the Knowledge-Based Libraries database, then it validates the two strings. The researcher considered the two strings match and returned zero as equivalent to the Levenshtein distance of zero, which is a 100% match result.

On the other hand, if the two strings have multiple values and have separated words in each column matching, the architectural design divides each of the words by space. In doing so, compare each of the divided words in the two-dimensional array that comes from the record in the Knowledge-Based Libraries database. If matches are found, one of the strings is considered as zero distance. Afterwards, average the string match to the Levenshtein function to detect the similarity of the left behind strings. This method helps the Levenshtein distance algorithm to detect similar words through knowledge-based and a series of rule-based like techniques. The two sequences of the architectural design are illustrated in Fig. 7.



Term1	Term2
St.	Street
Road	Rd.
7th	Seventh
No.	Number

Fig. 7: Sequences of the proposed method to enhance Levenshtein distance to improve duplicate record detection via Rule-Based Technique

III. Results and Discussions

A. Record Duplication Detection using Levenshtein Distance

No Blocking and Windowing Implemented in Record Duplication Detection

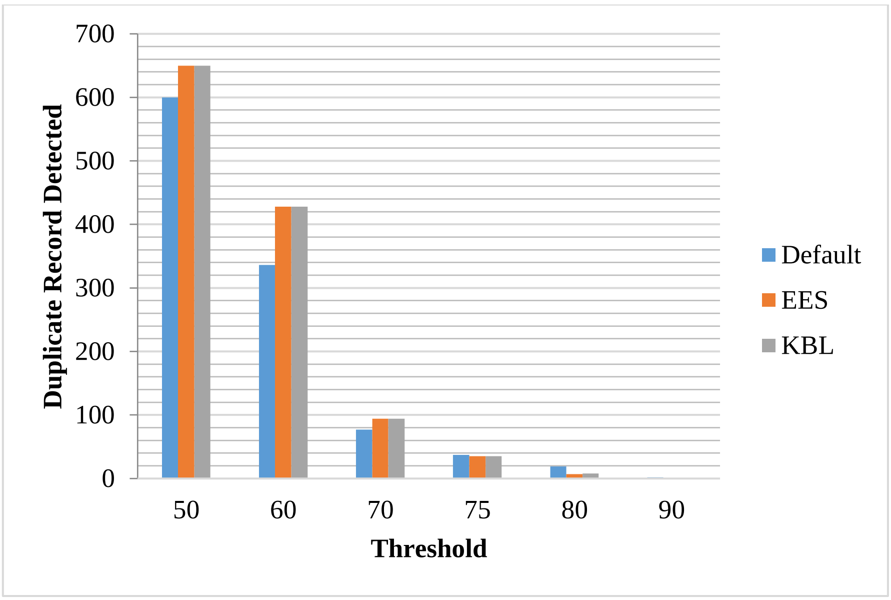


Fig. 8: Representation of the implementation of Levenshtein distance to detect duplicate records in the database

Fig. 8, illustrates the results of the implementation of the Levenshtein distance function to detect duplicate records in the database without blocking and windowing technique.

The chart has the following three implementation comparison: Default, which implemented the default implementation of Levenshtein function, the EES (Eliminating Equal String), which eliminated the equal string in a string before applying the Levenshtein function, and the KBL (Knowledge-Based Libraries) that helped the Levenstein function to determine knowledge-based comparison, for instance, St. for Saint and Street when abbreviated.

The graph was tested in a different threshold presented in 50, 60, 70, 75, 80, 90 percent to determine duplicate records in the database. The 50 percent threshold shows that more duplicate records were detected rather than the other thresholds. It took 600 duplicated rows of the default implementation, 650 records of the EES, and the KBL detected 650 duplicated records.

Whereas, the second test of a 60 percent threshold took an average of 397 detected records. But, the threshold of 70% only detected an average of 88 duplicated records. Based on the detailed result of the detected duplicated record, the most accurate threshold considered is the threshold of 75% and up are the most accurate comparison to 70% and below the threshold.

No Blocking and Windowing Implemented in Record Duplication Detection by Elapse Time

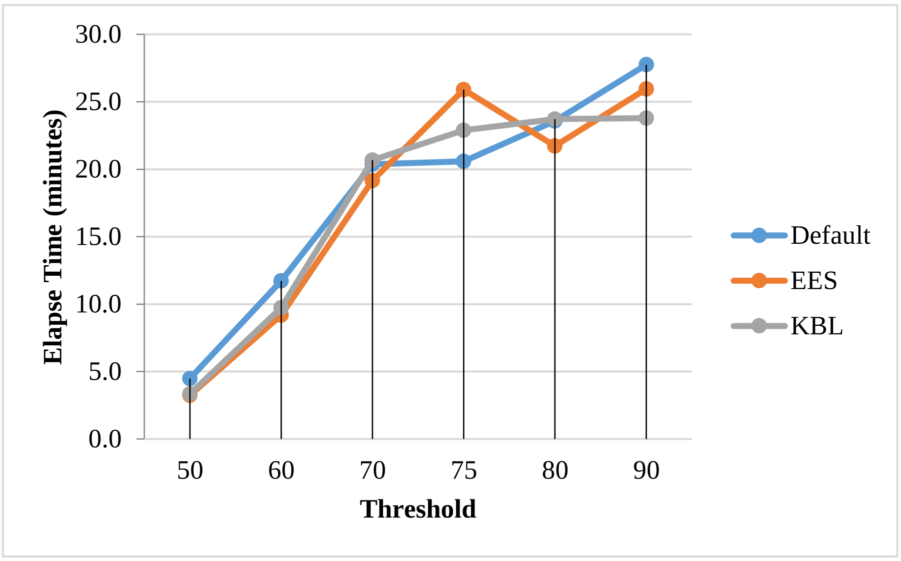


Fig. 9: Representation of the implementation of Levenshtein distance on the comparison elapse time per threshold

Fig. 9, displays the results of the elapsed time taken per threshold without implementing the blocking and windowing techniques. The results showed that the lower percentage of the threshold took faster to execute rather than the higher percentage of the threshold. However, as observed, without blocking and windowing technique, it consumed much time for the comparison of data. The total records of the testing datasets were only 864 records in one table.

The above presentation shows that the EES is the fastest to execute compared to the Default implementation and the Knowledge-based Libraries extension of the Levenshtein distance function.

Blocking and Windowing Implemented in Record Duplication Detection with a fixed size of five Partition

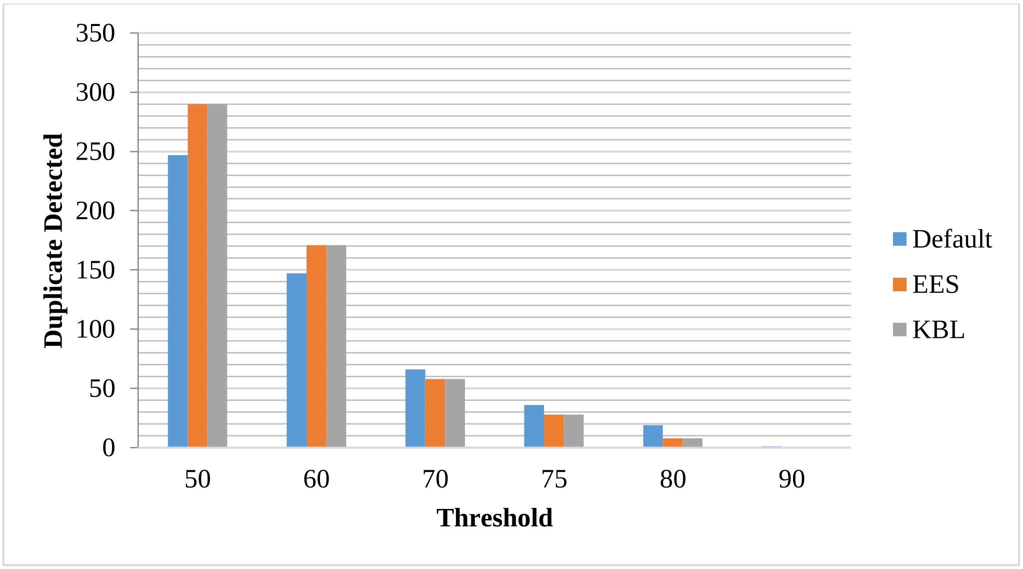


Fig. 10: Representation of the implementation of blocking and windowing techniques to detect duplicate records in the Sequel Database

Fig. 10, shows the result of the total duplicate records in implementing blocking and windowing techniques to speed-up the comparison of data. The results on each of the methods being tested are remarkable and had a difference of 43 records in the 50% threshold and 24 records difference in the threshold of 60%. In the threshold of 70%, the default method showed higher detected records compared to EES and KBL implementation methods up to the higher threshold.

Blocking and Windowing Implemented in Record Duplication Detection with a fixed size of five (5) Partition to speed-up data-comparison

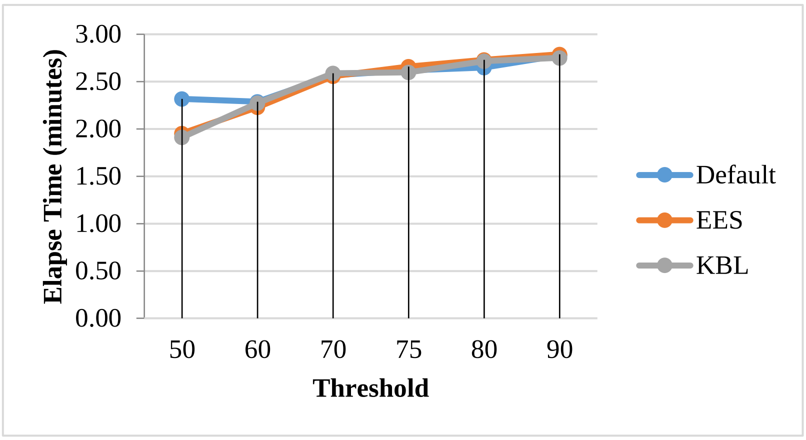


Fig. 11: Representation of the implementation of blocking and windowing techniques to optimize data-comparison to Sequel Database

Fig. 11, shows the outcome of the total amount of time that each approach took. The presentation results tell the effectiveness of the implementation of the blocking and windowing method. Due to the rapid speed of the comparison, it only takes all the methods more than two minutes execution rather than without partitioning techniques. It took more than 20 minutes before the comparison was executed.

IV. Conclusions

In summary, The researcher proposed new methods in record duplication detection through the Eliminating Equal String Method and the Knowledge-Based Method. It is concluded that the Eliminating Equal String Method and the Knowledge-Based Method could improve Levenstein Distance to identify real-world instances and terminologies.

The Levenshtein Distance Algorithm and the Rule-based Technique were employed by the researcher to increase the algorithm's capacity to find similarities between two strings and an equal string. Based on the outcome of testing in the Restaurant Dataset, data comparison was shortened to 18 minutes of execution with the aid of the Blocking and Windowing technique.

V. Recommendations

It is recommended that the methods should be tested on a high-end computing device and on a bigger dataset. This would further improve the data comparison execution that may affect the time, data cleaning, and detection.

VI. Future Works

Develop an application or tools that will implement the three (3) methods proposed to detect duplicate records in the database.

References

[1] V. P. Parmar and C. K. Kumbharana, “Comparing Linear Search and Binary Search Algorithms to Search an Element from a Linear List Implemented through Static Array, Dynamic Array and Linked List,” 2015.

[2] L. Yue, “The comparison of storage space of the two methods used in SQL server to remove duplicate records,” in *Procedia Computer Science*, Elsevier B.V., 2018, pp. 691–698. doi: 10.1016/j.procs.2018.04.313.

[3] B. Calabrese, “Data cleaning,” in *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, vol. 1–3, Elsevier, 2018, pp. 472–476. doi: 10.1016/B978-0-12-809633-8.20458-5.

[4] A. Skandar, M. Rehman, and M. Anjum, “An Efficient Duplication Record Detection Algorithm for Data Cleansing,” 2015.

[5] S. Rani and J. Singh, "Enhancing levenshtein's edit distance algorithm for evaluating document similarity," in *Communications in Computer and Information Science*, Springer Verlag, 2018, pp. 72–80. doi: 10.1007/978-981-13-0755-3_6.

[6] C. Zhao and S. Sahni, "String correction using the Damerau-Levenshtein distance," *BMC Bioinformatics*, vol. 20, Jun. 2019, doi: 10.1186/s12859-019-2819-0.

[7] R. A. Parazo and J. A. Esquivel, "Improving SQL query response time thru client side processing in client-server environment," in *Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering, WCSE 2019*, International Workshop on Computer Science and Engineering (WCSE), 2020, pp. 697–703. doi: 10.18178/wcse.2019.06.103.

[8] ABS-CBN News, "Gatchalian seeks urgent enforcement of nat'l ID system for effective contact tracing."

[9] Katrina Domingo, "Sotto: Implement national ID system to streamline COVID-19 aid recipients, speed up release," ABS CBN.

[10] Sheila Tejada, "Duplicate Detection, Record Linkage, and Identity Uncertainty: Datasets."

[11] S. Madnick, "P035".

[12] U. Draisbach, "IEEE Xplore Reference Download 2023.12.23.22.50.59".

[13] T. El-Shishtawy, "A Hybrid Algorithm for Matching Arabic Names."

[14] M. Rehman and V. Esichaikul, "Duplicate Record Detection for Database Cleansing," in *2009 Second International Conference on Machine Vision*, 2009, pp. 333–338. doi: 10.1109/ICMV.2009.43.

[15] S. R. Alenazi and K. Ahmad, "Record duplication detection in database: A review," *Int J Adv Sci Eng Inf Technol*, vol. 6, no. 6, pp. 838–845, 2016.

[16] M. A. Hern´, H. Andez, and S. J. Stolfo, "Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem," 1998.

[17] O. H. Akel and M. M. Aqel, *A Comparative Study of Duplicate Record Detection Techniques*. Middle East University, 2012.

[18] R. Baxter, P. Christen, and T. Churches, "A Comparison of Fast Blocking Methods for Record Linkage; erschienen in: Proceedings of the Workshop on Data Cleaning, Record Linkage and Object Consolidation at the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining," *Washington DC*, 2003.