An Efficient Duplication Record Detection Algorithm for Data Cleansing

Arfa Skandar
Lahore College for Women University,
Jail Road, Lahore, Pakistan

Mariam Rehman
Lahore College for Women University,
Jail Road, Lahore, Pakistan

Maria Anjum
Lahore College for Women University,
Jail Road, Lahore, Pakistan

ABSTRACT
The purpose of this research was to review, analyze and compare algorithms lying under the empirical technique in order to suggest the most effective algorithm in terms of efficiency and accuracy. The research process was initiated by collecting the relevant research papers with the query of "duplication record detection" from IEEE database. After that, papers were categorized on the basis of different techniques proposed in the literature. In this research, the focus was made on empirical technique. The papers lying under this technique were further analyzed in order to come up with the algorithms. Finally, the comparison was performed in order to come up with the best algorithm i.e. DCS++. The selected algorithm was critically analyzed in order to improve its working. On the basis of limitations of selected algorithm, variation in algorithm was proposed and validated by developed prototype.

After implementation of existing DCS++ and its proposed variation, it was found that the proposed variation in DCS++ producing better results in term of efficiency and accuracy. The algorithms lying under the empirical technique of duplicate records deduction were focused. The research material was gathered from the single digital library i.e. IEEE. A restaurant dataset was selected and the results were evaluated on the specified dataset which can be considered as a limitation of the research. The existing algorithm i.e. DCS++ and proposed variation in DCS++ were implemented in C#. As a result, it was concluded that proposed algorithm is performing outstanding than the existing algorithm.

General Terms
Data Quality, Data Cleansing, Dirty Data

Keywords
Duplication Records Detection Algorithm, DCS++, Windowing, Blocking

1. INTRODUCTION
Now-a-days, the digital economy is totally dependent on the databases. Many industries and businesses have huge amount of data stored in different databases. In this fast world, it is necessary that data operations on the database are carried out smoothly and efficiently [1]. However, to access the useful information that can help in decision making for industries and businesses, it is necessary to integrate large dataset. When data is integrated from different sources then it contains a huge part of dirty data. This dirty data contain mistakes in record values, duplication in records, spelling mistakes, null or illegal values, disobedience referential integrity and inconsistency in records [2].

Quality assurance of data is necessary for fast retrieval of data, quick and smooth data processing, and right decision making. Business organizations are paying high attention towards data quality because dirty data can effect important decisions in businesses. In addition, cleansed data can improve the production because of quality decisions [3]. Data cleansing is performed to get cleansed and quality data. Therefore, Data cleaning is important for business industry. The available data cleaning methods are not time and cost effective [4]. Duplication in data is one of the most important issues of Data cleaning. When data is gathered from different source then due to mistakes in spells or difference or inconsistency of format may cause presence of duplicate records in data [5]. Extraction of knowledge from huge databases is known as data mining [6]. Duplicate record deduction and data redundancy control are also hot topics of data mining and data integration [7,4]. With the increase of Quality data demand, many logical and statistical methods have been provided to resolve the problem[8]. In this regard, there are three basic techniques of Duplicate records detection which are knowledge-based techniques, probabilistic techniques and empirical techniques[3]. Many algorithms have been proposed under those techniques but all of them somehow lack in one of these parameters which are time efficiency, cost effectiveness, space consumption and accuracy [8]. Duplication record detection is a very diverse field so this decision was made that one of its basic technique will be chosen and then focus will be on algorithms which lie within that technique. It was decided to select empirical technique and compared all the algorithms under this category. After comparison, most effective algorithm will be selected and improved accordingly. The objectives of this research study are as follows:

1. To study the algorithms of duplication records detection
2. To perform comparative analysis of duplicate records detection algorithms lying under the empirical technique
3. To implement the best selected algorithm after performing comparative analysis
4. To suggest improvement in the selected algorithm

2. LITERATURE REVIEW
This section provides the necessary background material that is required to understand this research theme.

2.1 Types of Data Sources
Data can be retrieved from single and multiple sources. Therefore, data quality is ensured in both cases.

Single source data can have lack of integrity constraints and poor schema design at schema level and mistakes in data entries or duplication in data at instance level. Multi source

data faces the issue of numbering and structural conflicts at schema level and overlapping, contradiction, inconsistency of data at instance level[3].

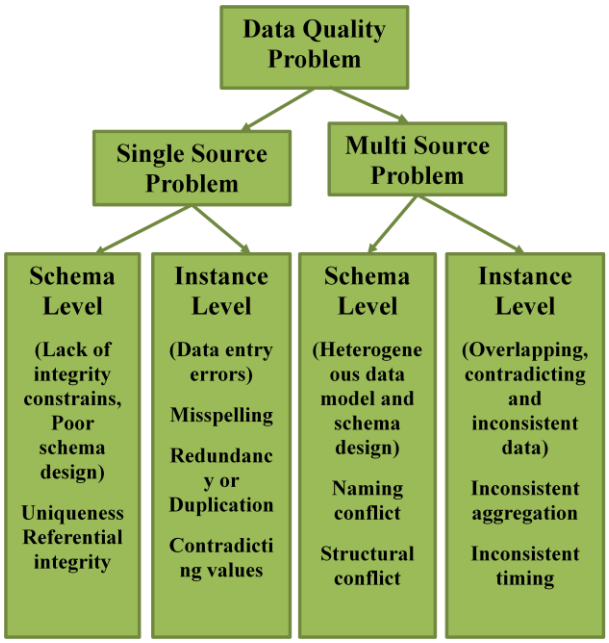

Fig1: Division of Data quality problems according to resources [3]

2.2 Dirty Data and Data Cleaning

When data is integrated from multiple sources then it contains a huge part of dirty data in it. This dirty data contain mistakes in records values, duplication in records, spelling mistakes, null or illegal values, disobedience referential integrity and inconsistency in records. This dirty data can infuse authentication of data. Therefore, it is necessary to clean data [2]. Data quality management is burning issue of enterprises because it has power to manipulate the decisions [9]. Therefore, data quality is spotted as bottleneck issue in businesses and industries [10].

Operational databases and online analytical processing systems cannot avoid the issue of data quality while integrating data. These issues are caused by non-unified set of standards in distributed databases. Data cleaning plays an important role in providing quality data by detection and removal of inconsistencies from data [11].

2.3 Duplicates and Types of Duplicates

Duplicates are the records that represent the same real-world object or entries. Record matching is a state of art technique to find these duplicates [12].

Duplicates can be of two types that are exact or mirror duplicates and approximate or near duplicates. Exact duplicate records contain the same content but on the other hand content of near duplicate records vary slightly [13]. The records which contain syntax differences or typographical errors but represent the same real world entity are known as near duplicates [14].

2.4 Duplication Records Detection and Types

Duplicate record detection is one of the most important data quality problems [15]. Detection of Duplicate plays an important role in record linkage, near duplicate detection and filtering queue [16]. Duplication detection is used to identify the same real world entities which exist in different format or representation in database [17,18]. It is very common to find some non-identical fields or records that refer the same entity [19]. Efficient and accurate detection of duplicates is hotspot of the data mining and online analyzer [4]. Now-a-day, duplication detection is the most popular topic in research [8]. Duplication detection is based on two basic Stages. The first one is the outer stage in which record matching technique or duplication record matching technique is applied. The second one is the inner stage that is based on field matching techniques.

Duplication record detection algorithms are divided in three types i.e. knowledge-based techniques, probabilistic techniques, and empirical techniques [3]. Empirical algorithms consist on sorting, blocking and windowing methods. Knowledge based algorithms demand training and the use of that training and reasoning skills in order to perform detection. Probabilistic algorithms are based on statistical and probability methods that are Bayesian networks, expectation maximization and data clustering. In this research study, focus is on empirical algorithms.

2.5 Empirical Algorithms

The general algorithms are as follows:

2.5.1 Blocking

Assign the *sorting key* to each record. Sort all the Records according to the key. Later, records are partitioned into disjoint partitions (means no record can be present in more than one partition) according to some *blocking key* (partition predicate).

Finally, comparison will be performed between records within the blocks. Using this technique, least comparisons will be performed [20].

2.5.2 Windowing

First of all, Merge two provided list of records. Sort all the records by *lexicon* order according to the attributes selected as a key. A fixed size slide window will be used. Records within the window will be compared with each other and first record will be released to select the next record in *fixed size* window.

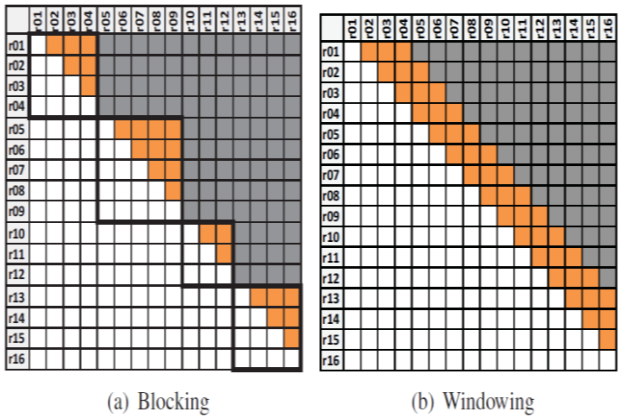

Fig2: Selection of elements for comparison in Windowing and Blocking [20]

2.6 Comparison among Windowing and Blocking techniques

Similarity and differences among these algorithms are discussed below:

Both algorithms try to perform reduction in number of record comparisons. For reducing comparisons, intelligent guesses are made about window / block sizes. In both algorithms, first of all data records are sorted and it is assumed that after sorting duplicates will be close to each other.

However, the mechanism of selection of records for comparisons is different from one another. In blocking algorithms, records are blocked in disjoint partitions. On the other hand, windowing algorithm works by sliding a window over the records [20].

The use of domain specific key for sorting can reduce the complexity of the algorithm but also cause domain dependency [21]. It is not even necessary to keep the key domain specific. Therefore, blocking and windowing methods such as sorted neighborhood are domain independent [22]. In this research, empirical algorithms are chosen due to the nature of domain independence.

2.7 Related Work

The algorithms have been discussed in detail below:

2.7.1 Sorted Blocks

Input Parameters: Records, key (may or may not be unique), overlapping value (o)

Records are blocked according to the partition predicate. After that records within the partition plus the overlapping records (Selected with the help of a fix size parameter) will be compared with each other.

Output: Duplicate or Non-Duplicates

2.7.2 Duplicate Count Strategy++

Input Parameters: *Records, Sorting key (key), Window Size (w), Threshold (ϕ)*

A growing window is slide over the records and records within the window are compared with each other. If a duplicate is found then it will be added to skipped list and will never be selected again for comparison which will ultimately reduce the number of comparisons.

Output: Duplicate or Non-Duplicates

2.7.3 Decision making algorithm

Input Parameters: Databases, Databases priorities values, Initial field priorities values, Initial threshold, Final threshold

Match the field count of each record and assign each field of first database to the field of other database. Set the priorities of fields and sort them accordingly. Select a specific number of fields of all records and compare them if any two records cross a specific threshold then these records will be compared further.

Output: Exact Similar, Approximate Similar, Less Similar and Non Similar

2.7.4 Nested Blocking

Input Parameters: Data source, standardization rules, blocking fields and Threshold

Records are divided into partitions then partitions are further divided into sub-partitions. Afterwards, comparison will be performed within sub-partitions.

Output: Duplicate, Possible Duplicate or Non-Duplicates

2.7.5 PC-Filter+

Input Parameters: Database, blocking key value, Size of partition (s), threshold (σ)

Records are blocked in equal size partitions. Records within the blocks will be compared. PCG (partition comparison graph) will be constructed for inter comparison. If number of blocks will be less than defined ratio then all blocks will be compared with each other. Otherwise, a defined number of neighboring blocks will be compared.

Output: Duplicate or Non-Duplicates

3. RESEARCH METHODOLOGY

The steps of research methodology adopted in this research study are shown in Fig 3 and their description is given below:

3.1 Set Aims and Objectives of Research

The main purpose of this research was to review different algorithms which have been proposed in the literature to suggest the most effective one in terms of efficiency and accuracy.

3.2 Preparation of Proposal

Some research articles were selected randomly from ACM and IEEE digital libraries. Based on these articles, proposal was written to defend and propose research topic.

3.3 Collection of Research Papers in the relevant domain i.e. duplicates records detection

Afterwards, it was decided that systematic review of literature will be followed. In order to perform the systematic review, different digital libraries were searched out for the research articles under duplication records detection keyword.

3.4 Search of research papers

While searching articles, it was found that IEEE digital library contains most relevant research articles. With the keyword of "duplicate records detection" total 61 articles were found.

3.5 Selection of relevant Research Papers and Division of Research Paper

Selected research articles were divided into four major categories. From these categories, there were three different techniques of duplication record detection and the articles which did not lie under these techniques were categorized as 'Others'.

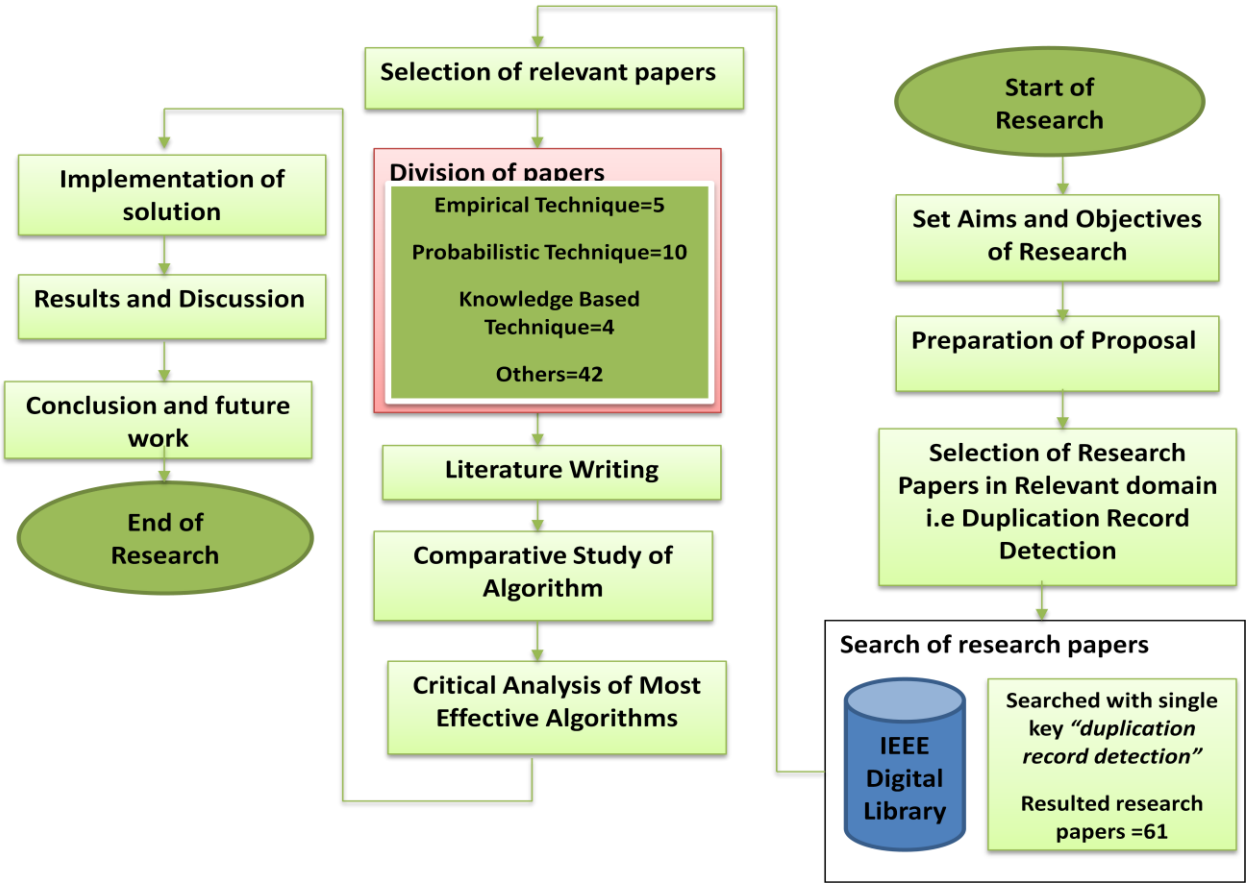

Figure 3: Research methodology

3.6 Literature Writing

Literature review was performed based on the selected articles and the main focus remained on empirical techniques and other techniques were ignored for the sake of this research.

3.7 Comparative Study of Algorithms under Empirical Technique

Comparative study of algorithms under the heading of empirical technique was performed in order to come up with comparative analysis. .

Critical analysis of most effective algorithm

Critical analysis of DCS++ was performed and suggestions were given for improvement of the algorithm.

3.8 Implementation of Solution

Solution is implemented for the existing DCS++ Algorithm and the proposed Algorithm.

3.9 Results and Discussions

The evaluation of algorithm was performed and the results have been discussed in detail below.

4. CRITICAL ANALYSIS DCS++

Windowing algorithm provides more accuracy instead of blocking. Therefore, DCS++ is selected because it is the most efficient windowing algorithm among all variants which are included in this study.

4.1 DCS++ Algorithm

Sort all the records according to the *sorting key*. Afterwards, put the *w* records in current window (*win*) sequentially. Now, select a record from all records sequentially and check whether the record is in skip records list (*SkipRecords*) or not. Compare the selected record with all the records within win and increase count of number of comparisons (*c*) by 1. If a record is found as a duplicate of Selected record then mark it as duplicate by adding it in to the *SkipRecords*, increase count of current duplicated record (*d*) by1and add the record in *win* sequentially till *win.lenght* < *duplicate record count* + *w-1* and *win.lenght with increase* < *records*. When all the records within the win are compared, remove the first record of *win*. If remaining records in *win* < *w* then add one record at the end Otherwise, remove records from the end till *win. Length* = *w* and move back to the step of selecting a record sequentially from all records. Continue the process till end of records [23].

4.2 Critical Point

Records are sorted according to single or composite key but not by all fields of the records. Therefore, it is possible that duplicate records lie in the same window but not consecutively. In Fig 4, full advantage of transitive property with DCS++ algorithm cannot be taken. It is clearly reflected by the Fig. 5, with any size of window, that record numbers 3, 4, 5, 6 even if add to the skip list in first window but will be compared again with record 2 in second window.

In such case, DCS++ will perform unnecessary comparisons. As shown in Fig 4. The problem can be resolved by increasing a single check in the algorithm. After that, algorithm will avoid those unnecessary comparisons.

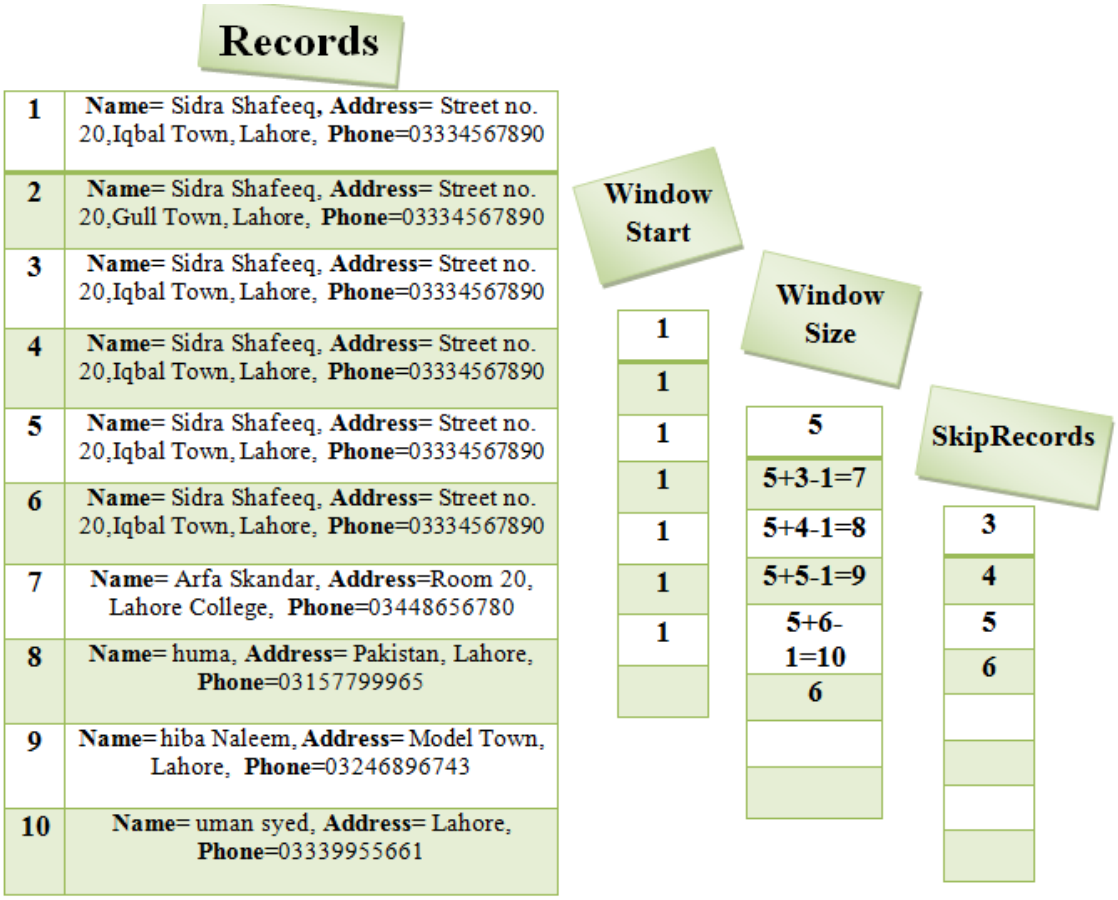

Fig4: Working of DCS

4.3 Proposed Algorithm

Sort all the records according to the *sorting key*. Afterwards, put the *w* records in current window (*win*) sequentially. Now, select a record from all records sequentially and check whether the record is in skip records list (*SkipRecords*) or not. Compare the selected record with all the records within *win* that are not in *SkipRecords* and increase count of number of comparisons (*c*) by 1 with each comparison. If a record is found as a duplicate of Selected record then mark it as duplicate by adding it in to the *SkipRecords*, increase count of current duplicated record (*d*) by1and add the record in win sequentially *till win.lenght < duplicate record count + w-1* and *win.lenght with increase < records*. When all the records within the win are compared, remove the first record of win. If remaining records in *win < w* then add one record at the end. Otherwise, remove records from the end till *win. Length = w* and move back to step of selecting a record sequentially from all records. Continue the process till end of records.

Now, the prototype of proposed algorithm is developed to find that whether with the improvement in DCS++ have retained its accuracy. Secondly, an attempt is being made to see whether with a good String matching algorithm, is there any potential to have higher precision value.

5. EVALUATION METHODOLOGY

The evaluation of algorithm is performed to compute the accuracy and efficiency of the algorithms. The accuracy refers to the number of duplicate detected and the efficiency refers to the number of comparisons performed to detect those duplicates.

5.1 Evaluation Parameters

The correct detection of duplicate is true positive (TP). When a record is not duplicate but detected as duplicate then it will be called false positive (FP). False negatives (FN) are the records that are duplicate but not detected as duplicate. Moreover, numbers of comparisons have been performed under a single run of algorithm in order to see the complexity of algorithm.

Existing algorithm of DCS++ and the proposed algorithm are evaluated for accuracy on the basis of formulas defined in Table 1.

Table 1. Quality Measures [3]

Quality Measure	Formula
Precision	TP/(TP+FP)
Recall	TP/(TP+FN)
F-Score	2*(Precision * Recall)/(Precision + Recall)

The efficiency is measured on the basis of number of comparisons performed during the record comparisons.

5.2 Dataset

Dataset selection for evaluation is an important task. To evaluate algorithms, it was decided to use benchmarked data.

5.3 Restaurant Database

A database of a Restaurant is selected for the purpose of evaluation [3]. The attributes of database are Name, Address and City. There are 865 records in the database.

5.4 Detection of Duplicates for Evaluation

For the purpose of evaluation, duplicates are marked in dataset manually. After detection, 202 matching pairs were found but 12 out of these require knowledge base. These 12 records are matched by Name and address but in City field if one record contain the city name then the other record contain district or the closest famous city. Therefore, these records are excluded from set of duplicates in order to perform evaluation. So, there are 190 records left which means 95 duplicates. During the detection of duplicates, order remained as City, Address, and Name respectively. In the original source, numbers of duplicates mentioned are 112 but in this research, they are 95. The error percentage is ((Numbers of duplicates actually exist - Numbers of duplicates detected)/Number of total records in dataset)*100.

Error Percentage = ((112-95)/865)*100=1.9675%

The error percentage is extremely low. Therefore, it is negligible.

6. EVALUATION OF ALGORITHMS

In this section, algorithms are evaluated with respect to the parameters defined above.

6.1 DCS++ with Exact Matching

After conducting experimental evaluation of DCS++ algorithm, with naïve matching algorithm at field level, results are described in Table 2. The window size is set to be 6 for evaluation.

Table 2. Evaluation of DCS++ with Modified Naïve String Matching Algorithm

Quality Measures	Resulted Values
True Positives	69
False Positives	0
False Negatives	26
Precisions	100%
Recall	72.63%
F-Score	84.14%
Number of Comparisons	3244

Table 2 shows that the results of DCS++ with Naïve Exact String Matching algorithm are not bad. The numbers of false positive are 0 but numbers of True Positives and recall values are extremely low.

6.2 Proposed Algorithm with Exact String Matching

After conducting experimental evaluation of proposed algorithm, with naïve matching at field level, results are described in Table 3. The selected window size for evaluation is 6.

Table 3.Evaluation of Proposed Algorithm with Modified Naïve String Matching Algorithm

Quality Measures	Resulted Values
True Positives	69
False Positives	0
False Negatives	26
Precisions	100%
Recall	72.63%
F-Score	84.14%
Number of Comparisons	3244

Table 3 shows that the results of proposed algorithm with Naïve Exact String Matching algorithm are same as DCS++ , but there is no improvement in number of comparisons. On the other hand, there is no negative effect on the results with the change in algorithm.

6.3 Approximate String Match

It is not enough to check the algorithm with exact string matching only. In order to see the effect of change in algorithm with approximate string matching algorithm and to find that proposed string matching algorithm is helping in order to improve results of DCS++ or not.

6.3.1 DCS++ with Basic String Matching

DCS++ with modified Basic String Matching Algorithm is used for approximate string matching. The evaluation is performed by ranging the threshold value from 0.45 to 0.65 with the gap of 0.05. The results are shown in Table 4.

Table 4. Evaluation of DCS++ with Modified Basic String Matching Algorithm

DCS++ Algorithm with Basic String Matching Algorithm (Modified)								
S. NO	**Threshold**	**TP**	**FP**	**FN**	**Precisions (%)**	**Recall (%)**	**F-Score (%)**	**NC**
1	0.45	86	17	9	83.50	90.53	86.87	3166
2	0.5	86	13	9	86.87	90.53	88.66	3168
3	0.55	80	3	15	96.39	84.21	89.89	3204
4	0.6	78	3	17	96.30	82.11	88.64	3210
5	0.65	76	2	19	97.44	80.00	87.86	3219

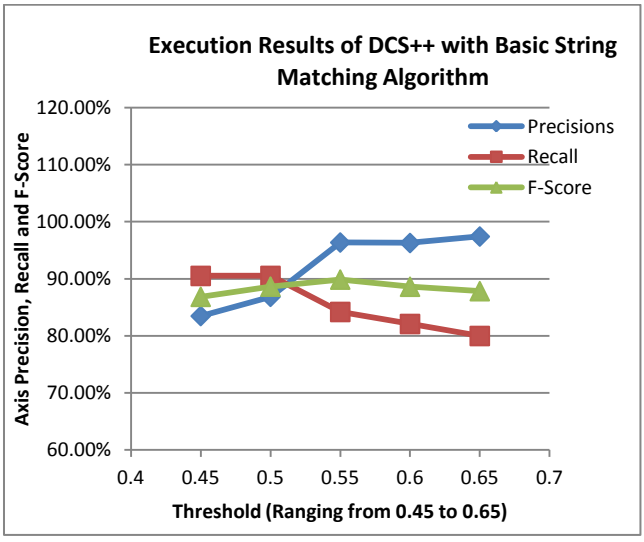

Fig 5: Execution Results of DCS++ with Modified Basic String Matching Algorithm

Fig 5 is representing the relationship between F-Score, Precision and Recall. It shows that there is a reverse relationship between Precision and Recall values for DCS++ with Basic String Matching Algorithm. On the other hand, F-Score has a positive relationship with both Precision and Recall.

6.3.2 Proposed Algorithm with Basic String Matching

The proposed with modified Basic String Matching Algorithm is used for approximate string matching. The evaluation is performed by ranging the threshold from 0.45 to 0.65 with the gap of 0.05. The results are shown in Table 5.

Table 5. Evaluation of Proposed Algorithm with Modified Basic String Matching Algorithm

Proposed Algorithm with Basic String Matching Algorithm (Modified)								
S. NO	**Threshold**	**TP**	**FP**	**FN**	**Precisions (%)**	**Recall (%)**	**F-Score (%)**	**NC**
1	**0.45**	86	17	7	83.50	92.47	87.76	3144
2	**0.5**	86	13	9	86.87	90.53	88.66	3154
3	**0.55**	80	3	15	96.39	84.21	89.89	3202
4	**0.6**	78	3	17	96.30	82.11	88.64	3208
5	**0.65**	76	2	19	97.44	80.00	87.86	3217

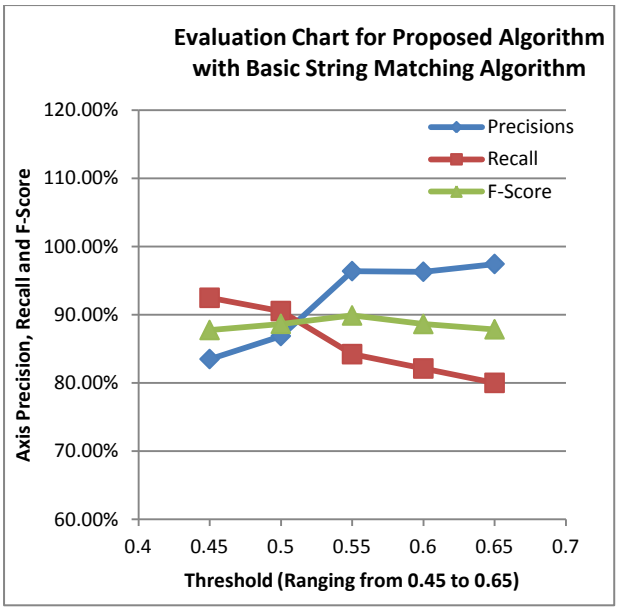

Fig 6: Execution Results of Proposed with Modified Basic String Matching Algorithm

Fig 6 describes that if the value of precision is increasing then the value of Recall is decreasing. Therefore, it is an inverse relationship. On the other hand, F-Score have closer values which show the positive relationship of F-Score with both recall and precision.

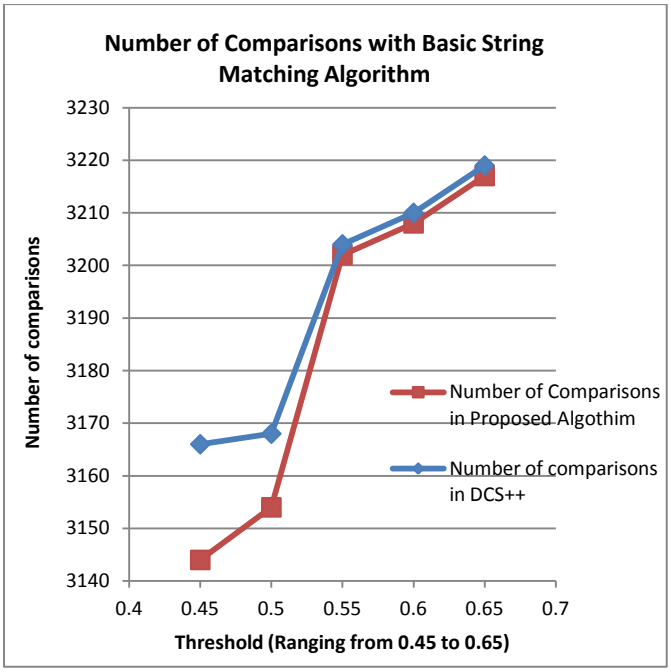

Fig 7: Number of Comparisons performed by DCS++ and Proposed Algorithm with Modified Basic String Matching Algorithm

Fig 7 shows that the numbers of comparisons required in Proposed Algorithm with Basic String Matching Algorithm are less than of the DCS++ Algorithm.

6.3.3 DCS++ with Recursive String Matching

DCS++ with modified Recursive String Matching Algorithm is used for approximate string matching. The evaluation is performed by ranging the threshold value from 0.45 to 0.65 with the gap of 0.05. While running DCS++, the most accurate results were gained at 0.65. After that, with higher threshold value than 0.65, the recall decreases. The results are described in Table 6.

Table 6. Evaluation of DCS++ with Modified Recursive String Matching Algorithm

DCS++ with Recursive String Matching Algorithm (Modified)								
S. NO	**Threshold**	**TP**	**FP**	**FN**	**Precisions (%)**	**Recall (%)**	**F-Score (%)**	**NC**
1	0.45	91	46	4	66.42	95.79	78.45	3094
2	0.5	92	45	3	67.15	96.84	79.31	3094
3	0.55	91	4	4	95.79	95.79	95.79	3169
4	0.6	91	4	4	95.79	95.79	95.79	3169
5	0.65	91	4	2	95.79	97.85	96.81	3174

The use of modified form of Recursive String Matching Algorithm with DCS++ takes the precision to 95.79%, Recall to 97.85% and the F-Score to 96.81%. The higher level of accuracy is achieved with the threshold value of 0.65. With the threshold values which are greater than 0.65, the values of Recall, Precision, and F-Score decreases.

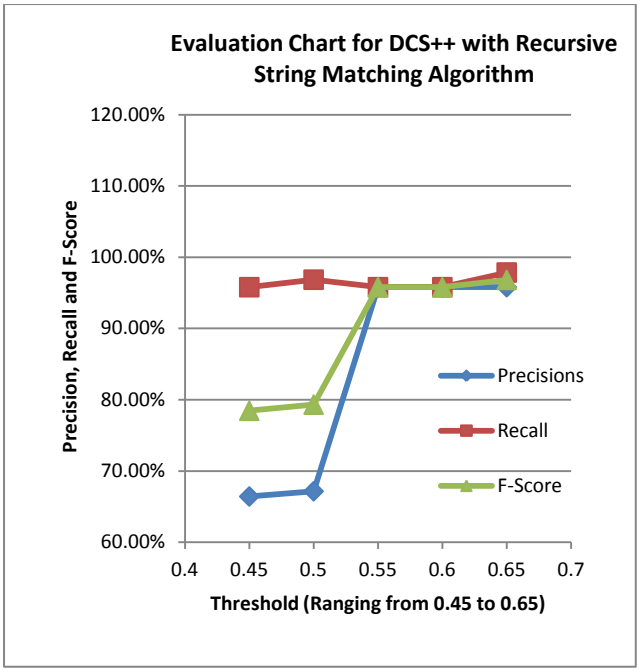

Fig 8: Execution Results of DCS++ with Modified Recursive String Matching Algorithm

Fig 8 shows that with lower threshold value precision is very low, but with efficient choice of threshold value, high precision and recall is achieved.

6.3.4 Proposed Algorithm with Recursive String matching

Proposed Algorithm with modified Recursive String Matching Algorithm is used for approximate string matching. The evaluation is performed by ranging the threshold value from 0.45 to 0.65 with the gap of 0.05. While running DCS++, the most accurate results were gained at 0.65. After that, with higher threshold value, the recall decreases. The results are described in Table 7.

Table 7. Evaluation of Proposed Algorithm with Modified Basic String Matching Algorithm

Proposed Algorithm with Recursive String Matching Algorithm (Modified)								
S. NO	**Threshold**	**TP**	**FP**	**FN**	**Precisions (%)**	**Recall (%)**	**F-Score (%)**	**NC**
1	0.45	92	41	3	69.17	96.84	80.70	3033
2	0.5	93	40	2	69.92	97.89	81.58	3033
3	0.55	91	4	4	95.79	95.79	95.79	3166
4	0.6	91	4	4	95.79	95.79	95.79	3166
5	0.65	91	4	2	95.79	97.85	96.81	3172

The use of Proposed Algorithm with modified form of Recursive String Matching Algorithm takes the precision to 95.79%, recall to 97.85% and the F-Score to 96.81%. The higher level of accuracy is achieved with the threshold value of 0.65.

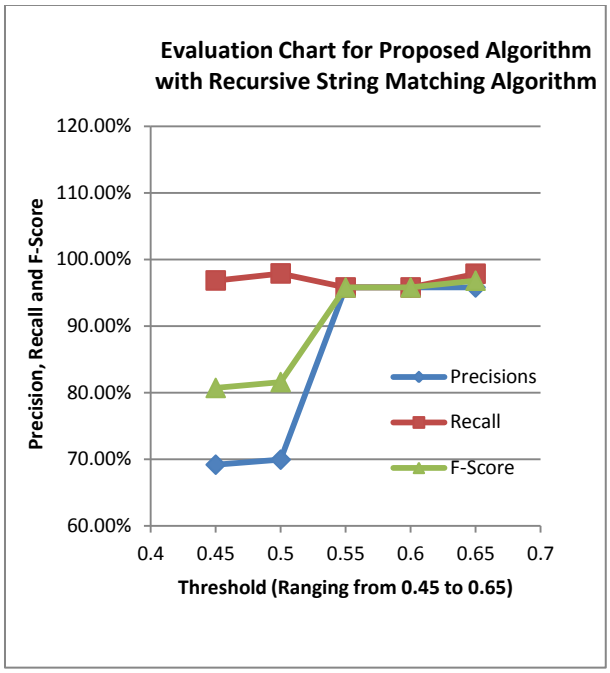

Fig9: Execution Results of Proposed with Modified Recursive String Matching Algorithm

Fig 9 shows that with the right choice of threshold value, proposed Algorithm with Recursive String Matching algorithm can achieve higher precision, recall and F-Score values.

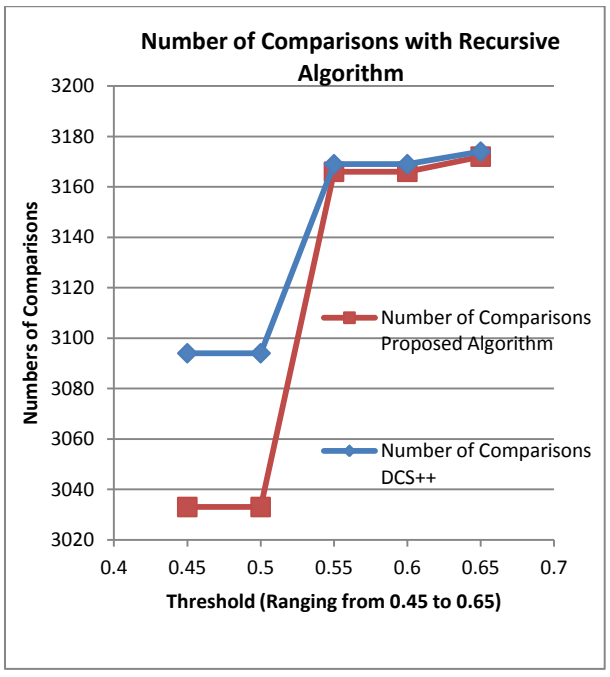

Fig 10: Number of Comparisons performed by DCS++ and Proposed Algorithm with Modified Recursive String Matching Algorithm

Fig 10 shows that the numbers of comparisons in Proposed Algorithm with Recursive String Matching Algorithm are less with lower threshold value than of the DCS++ Algorithm.

7. RESULTS AND DISCUSSIONS

It is concluded by Table 2 and Table 3 that with the use of exact string matching algorithm, the accuracy of proposed algorithm is same as accuracy of DCS++ algorithm and there is no improvement in number of comparisons. The results are not bad as 100% Precision and 70.63% Recall is achieved. The most noticeable thing is that, there is not a single false detection of duplicate with naïve string matching algorithm. Overall, results with naïve algorithm are satisfactory.

The Proposed Algorithm with Basic String Matching Algorithm requires reduced number of comparisons instead of DCS++ with Basic String Matching Algorithm. On the other hand, both algorithms have same accuracy. By Table 2, 3, 4 and 5, it can be concluded that the Recall value of both algorithms i.e. DCS++ and Proposed Algorithm with Basic String match algorithm by using the right threshold value is more than of naïve algorithm, but this gain requires the little compromise on the Precision value.

Table 6 and 7 shows that the Proposed Algorithm that is implemented with the modified Recursive Algorithm is performing more accurately and efficiently than of DCS++ with Recursive Algorithm with lower threshold values but with higher threshold values they have same performance. Another important aspect is the gain of 96.81% F-Score value. It can be concluded by taking look at Table 4, 5, 6 and 7 that DCS++ and Proposed Algorithm with Recursive Algorithm is performing much better than of Basic String matching algorithm, but the error percentage of DCS++ and Proposed algorithm with best F-Score is ((Numbers of duplicates actually exist - Numbers of duplicates detected)/Number of total records in dataset)*100= ((112-91)/865)*100=2.43%. This error percentage is extremely low so it is negligible.

With the help of above discussion, it can be concluded that the proposed algorithm which is implemented with the help of Modified Recursive Algorithm outperforms than of all other algorithms in term of accuracy and efficiency.

8. CONCLUSION

The most challenging task of this research study was to prove that after making changes in the basic algorithm of the DCS++, there is no loss of efficiency or accuracy instead of proving the improvement. Prototype of both original DCS++ algorithm and the new proposed algorithm is implemented. With the results of evaluation, it is concluded that with Exact String or Field match both algorithms work almost in similar manner. On the other hand, with Approximate String or Field match number of comparisons are reduced by the proposed algorithm.

Moreover, accuracy in terms of recall, precision and F-Score is almost similar for both algorithms, but in case where Proposed Algorithm is used with modified recursive algorithm with minimum threshold value, it produces more accurate results than of original DCS++.

It is also proved that it is mostly not possible in case of real data that all duplicates are detected with the use of exact string matching algorithm, even if the precision reached to 100% but the F-Score is lower. The reason of using two different approximate algorithms was to show that there is a room to gain higher rate of duplicate detection with the same record detection algorithm by using more efficient string matching algorithm. The recursive algorithm is used by calling twice for a single string match to gain high accuracy with a non-symmetrical algorithm. It increases the complexity but outperforms with the efficient choice of the threshold value.

The proposed algorithm is the best choice for the task of duplication record detection. It is domain independent but input dependency is there. The algorithm provides almost similar results than of DCS++ in terms of accuracy excluding some cases where accuracy of proposed algorithm is higher. On the other hand, efficiency of proposed algorithm is equal or higher in some cases.

9. RECOMMENDATIONS AND FUTURE WORK

The research is scoped to the empirical techniques only. However, other techniques can be explored with the same directions. For the approximate string matching, basic and recursive algorithms are improved and applied to see their effects. There are many other algorithms which exist for approximate string matching and yet 100% precision and recall with the approximate match is not achieved yet so other techniques can also be applied to produce better results. Moreover, Window can be slide on field along with records instead of sliding window only on the records. This means that instead of selecting only number of records in a window, the number of fields can also be reduced with respect to its important.

In this research, it was found that there exist some records which require knowledge base to detect duplicates correctly. For example, Arfa Sikander, Street number 5 iqbal roads, Daska and Arfa Sikander, Street number 5 iqbal road, Sialkot are the same records but in the first case nearby famous city name is mentioned. Moreover, there are also few other cases which are not being handled by the recursive algorithm. For example, 7th or seventh, these both cases cannot be handled without knowledge base.

10. REFERENCES

[1] Ahmed K. Elmagarmid, P., G. Ipeirotis, and Vassilios S. Verykios, "Duplicate Record Detection: A Survey," IEEE Trans. on Knowl. and Data Eng., vol. 19, pp. 1-16, 2007.

[2] P. Ying, X. Jungang, C. Zhiwang, and S. Jian, "IKMC: An Improved K-Medoids Clustering Method for Near-Duplicated Records Detection," in Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on, Wuhan, 2009, pp. 1 - 4.

[3] M. Rehman and V. Esichaikul, "DUPLICATE RECORD DETECTION FOR DATABASE CLEANSING," in Machine Vision, 2009. ICMV '09. Second International Conference on , Dubai, 2009 , pp. 333 - 338.

[4] X. Mansheng, L. Youshi, and Z. Xiaoqi, "A PROPERTY OPTIMIZATION METHOD in SUPPORT of APPROXIMATELY DUPLICATED RECORDS DETECTING," in Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on, 2009.

[5] Q. Hua, M. Xiang, and F. Sun, "An Optimal Feature Selection Method for Approximately Duplicate Records," in Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on, Chengdu, 2010.

[6] D. Bhalodiya, M., K. Patel, and C. Patel, "An Efficient way to Find Frequent Pattern with," in Nirma University International Conference on Engineering, 2013.

[7] L. Huang, P. Yuan, and F. Chu, "Duplicate Records Cleansing with Length Filtering and Dynamic Weighting," in Semantics, Knowledge and Grid, 2008. SKG '08. Fourth International Conference on, Beijing, 2008, pp. 95 - 102.

[8] M. Gollapalli, X. Li, I. Wood, and G. Governatori, "Approximate Record Matching Using Hash Grams," in 11th IEEE International Conference on Data Mining Workshops, 2011.

[9] Z. Wei, W. Feng, and L. Peipei, "Research on Cleaning Inaccurate Data in Production," in Service Systems and Service Management (ICSSSM), 2012 9th International Conference on, Shanghai, 2012.

[10] L. Zhe and Z. Zhi-gang, "An Algorithm of Detection Duplicate Information Based on Segment," in International Conference on Computational Aspects of Social Networks, 2010.

[11] H., H. Shahri and Z., A., A. Barforush, "Data Mining for Removing Fuzzy Duplicates Using Fuzzy Inference," in Processing NAFIPS '04. IEEE Annual Meeting of the (Volume:1), 2004.

[12] W. Su, J. Wang, and H., F. Lochovsky, "Record Matching over Query Results from Multiple Web Databases," in IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 2010.

[13] R. Naseem, S. Anees, M., and S. Farook, "Near Duplicate Web Page Detection With Analytic Feature Weighting," in Third International Conference on Advances in Computing and Communications, 2013.

[14] L., Wan Zhao and Wah, C. N., "Scale-Rotation Invariant Pattern Entropy for Keypoint-Based Near-Duplicate Detection," in IEEE TRANSACTIONS ON IMAGE PROCESSING, 2009.

[15] G. Beskales, A., M. Soliman, F., I. Ilyas, S.i Ben-David, and Y. Kim, "ProbClean: A Probabilistic Duplicate Detection," in Data Engineering (ICDE), 2010 IEEE 26th International Conference on, 2010.

[16] J. Kim and H. Lee, "Efficient Exact Similarity Searches using Multiple," in IEEE 28th International Conference on Data Engineering, 2012.

[17] M. Ektefa, F. Sidi, H. Ibrahim, and M.,A. Jabar, "A Threshold-based Similarity Measure for Duplicate Detection," in Open Systems (ICOS), 2011 IEEE Conference on, Langkawi, 2011, pp. 37 - 41.

[18] M. Herschel, F. Naumann, S. Szott, and M. Taubert, "Scalable Iterative Graph Duplicate Detection," in IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 2012.

[19] Q. Kan, Y. Yang, S. Zhen, and W. Liu, "A Unified Record Linkage Strategy for Web Service," in Third International Conference on Knowledge Discovery and Data Mining, 2010.

[20] U. Draisbach and F. Naumann, "A Generalization of Blocking and Windowing Algorithms for Duplicate Detection," in Data and Knowledge Engineering (ICDKE), 2011 International Conference on , Milan, 2011, pp. 18 - 24.

[21] A. Bilke and F. Naumann, "Schema Matching using Duplicates," in Proceedings of the 21st International Conference on Data Engineering, 2005.

[22] Q. kan, Yan, Y. g, W. Liu, and X. Liu, "An Integrated Approach for Detecting Approximate Duplicate Records," in Second Asia-Pacific Conference on Computational Intelligence and Industrial Applications, 2009.

[23] U. Draisbach, F. Naumann, S. Szott, and O. Wonneberg, "Adaptive Windows for Duplicate Detection," in 28th International Conference on Data Engineering, 2012.