DUPLICATE RECORD DETECTION IN DATA CLEANING USING DCS++ ALGORITHM

YIN YIN PHYO

M.C.Sc. **JUNE 2021**

DUPLICATE RECORD DETECTION IN DATA CLEANING USING DCS++ ALGORITHM

By

YIN YIN PHYO

B.C.Sc.

A dissertation submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (M.C.Sc.)

University of Computer Studies, Yangon
JUNE 2021

ACKNOWLEDGEMENTS

First of all, I wish to express my deepest gratitude and sincere appreciation to all persons, who contributed directly or indirectly towards the achievement of completing the thesis and helped me throughout the period of studies in University of Computer Studies, Yangon.

As being one of the first of these persons, I would like to express my appreciation and sincere thanks to **Prof. Dr. Mie Mie Thet Thwin**, the Rector, the University of Computer Studies, Yangon, for her kind permission to submit this thesis and general guidance, workable environment during the period of study.

I would like to extend gratitude to **Prof. Dr. Yadana Thein**, Pro Rector, the University of Computer Studies, Yangon, for her kind permission to submit this thesis.

My heartfelt thanks and respect go to **Dr. Thi Thi Soe Nyunt**, Professor and Head of Faculty of Computer Science, University of Computer Studies, Yangon, for her administrative support and invaluable guidance, as Dean of Master Course, throughout the development of the thesis.

I also deeply thank **Daw Aye Aye Khine**, Associate Professor, Department of English, the University of Computer Studies, Yangon, for editing my thesis from the language point of view.

A special word of gratitude is due to my supervisor, **Dr. Thidar Win,** Lecturer, Software Department, the University of Computer Studies, Yangon, for her invaluable recommendations regarding the thesis topic, giving me detailed guidance throughout the work of this thesis and support for my thesis work.

In addition, I would like to thank the board of examiners for making precious comments and detailed corrections to my thesis and those who are pressing power to improve the end result.

Last but not least, I am grateful to my family who has provided fully emotional or physical support throughout my student life. And then, I especially thank all my teachers for their valuable advice, opinions and participation in the seminars. Lastly, I also would like to thank my colleagues, friends and staff of University of Computer Studies, Yangon for providing necessary information, documentation requirements and collaboration during the seminars.

STATEMENT OF ORIGINALITY

I hereby certify that the work embodies in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

---------- ----------

Date Yin Yin Phyo

ABSTRACT

Nowadays, Duplicate Record Detection is a multiple record search process that represents the same physical entity in a dataset. It is also known as the record linkage (or) entity matching. The databases contain a very large dataset. Datasets contain duplicate records that do not share a common key or contain errors such as incomplete information, transcription errors and missing or differing standard formats (non-standardized abbreviations) in the detailed schemas of records from multiple databases. Therefore, the duplicate detection needs to complete its process in a very shorter time. Duplicate detection requires an algorithm for determining whether records are duplicate records or not.

In this system, the researcher calculates a similarity metric that is commonly used to find similar field items and uses the Duplicate Count Strategy-Multi Record Increase (DCS++) Algorithm for approximately duplicate records detection over publication xml dataset.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

CHAPTER 1

INTRODUCTION

Nowadays, the amount of data within the data warehouses becomes more and more huge and data errors or inconsistent data in these data warehouses also grow rapidly as the technology advances. In the economic world, invalid and duplicate data can be costly because it can affect the key decisions for operations in many industries and the production of business organizations. Therefore, data needs to be good quality. In order to improve data quality, data cleaning is especially necessary when integrating disparate data sources [1]. By integrating data from different sources and implementing a data warehouse, organizations become aware of possible differences and systematic conflicts.

The problem of identifying duplicate records in the database is an important step in the data cleanup and integration process. Data reduction is the process of detecting and removing data errors, inconsistencies, and duplicate data. Duplicate detection is one of the solutions of data cleaning. It has two tasks to detect duplicate records efficiently and effectively:

- the representation of the data may vary slightly, so a specific similarity measure needs to be defined to compare pairs of records and
- not all records can be peer compared because the data set may be large. To perform the task two, a number of algorithms have been proposed that split the dataset and compare all pairs of records in each partition.

Sorted Neighborhood Method (SNM) is a known way to advance the window by classifying data based on the sorting key and comparing only the records displayed in the same window. This paper proposes the Duplicate Count Strategy-Multi Record Increase Approach (DCS++), a variation of SNM and improvement of Duplicate Count Strategy (DCS). If a duplicate is found on the sorted dataset, it can also detect the other possible duplicates by comparing the next w - 1 record of that duplicate. It can also reduce the comparing time by skipping windows for duplicates. Therefore, the proposed system can be faster and can detect more duplicate records.

1.1 Motivation

As the technology advances, every dataset contains some errors, incomplete information and unstandardized formats. Every analyst experiences in wasting time for wrong conclusions because of these errors. And the time needed for analysis is typically spent in “cleaning” the data. In the business world, incorrect data can be costly in queries time and storage space for large scale databases. The duplicated records can cause incorrect results in analysis queries and erroneous data mining model to be built. The problem of detecting and eliminating duplicated data is one of the major problems in the broad area of data cleaning and data quality. To remove duplicate records from a dataset, the main consideration is how to decide that two records are duplicated. Records are compared to determine their degree of similarity, which implies that corresponding fields in the records has to be compared.

1.2 Related Works

Many researchers do research on duplicate record detection with different efficient and effective blocking and windowing methods [2].

Ying Pei et al. [3] implemented the K-medoids clustering algorithm (IKMC) to solve the problem of detecting almost duplicate records. It is considered as one separated data object for every record in the database. It uses the Edit Distance method to get similarity values between records. Finally, clustering of these similarity values can detect duplicate records. The algorithm can automatically adjust the number of clusters by comparing the similarity value with a predefined similarity threshold. This algorithm shows good detection accuracy and high availability.

Qiaoqiao Yang et al. [4] implemented the SNM algorithm based on some edit distances and variable windows to solve the shortcomings of the SNM algorithm. The algorithm proposed in this paper is based on the various edit distances and variable windows. The experiment's data set comes from the refrigeration industry management system. This proposed algorithm can efficiently recognize duplicate big data records. However, there is still the problem of improving the recall ratio and handling non-standard samples.

Jumoke Soyemi et al. [5] implemented a system for detecting duplicate records in a database using a simil matching algorithm. The Simil algorithm is based on

calculating the similarity between two strings. This proposed system can only be used to clean up data and prevent incorrect data from accessing the database.

1.3 Objectives of the Thesis

The main objectives of the thesis are as follows:

- To study the concepts of data cleaning
- To learn empirical methods such as sorted neighborhood method and its improvement methods
- To apply Duplicate Count Strategy-Multi Record Increase (DCS++) algorithm with Levenshtein Distance Algorithm of field matching techniques
- To provide the duplicates identification by using Adaptive DCS++ method
- To explore the effective and efficient method on duplicates detection process

1.4 Organization of the Thesis

This thesis is mainly composed of five chapters.

Chapter 1 is the introductory section where the introduction of duplicated records detection, the related works, the objectives and the organization of the thesis are presented.

Chapter 2 describes the background theory related to this thesis such as data quality, data cleaning, record matching problem, field matching techniques, record matching techniques and the duplicate records detection approaches.

Chapter 3 presents the design of the proposed system that is described as the system flow, description about Cora publication dataset that is used, the detail steps of parsing, standardization in preprocessing, the field similarity measuring using Edit Distance or Levenshtein Distance algorithm, the detection of duplicate record pairs using Duplicate Count Strategy-Multi Record Increase (DCS++) algorithm and performance evaluation.

Chapter 4 mainly describes the implementation of the proposed system in detail that includes the experimental setup, the system's implementation, the process of

detecting duplicate record pairs from xml dataset using DCS++ algorithm and the experimental result.

Finally, Chapter 5 concludes this thesis by highlighting the limitations and further improvement works of the proposed system.

CHAPTER 2

BACKGROUND THEORY

Nowadays, the word ‘data’ is mostly used to talk about the facts that are kept and shared electronically in databases. Quality of data is critical in getting to final analysis. Most of the organizations pay attention to the good quality of data for making business decisions. Data cleaning is one of the main processes of data preparation and correcting the inaccurate records from a record set, table or database. This chapter describes data quality, overview of data cleaning, the techniques of field matching with string data, record matching techniques and the several approaches of duplicate records detection process.

2.1 Data Quality

Data are abstract representations of chosen characteristics of real-world objects such as people and places, etc. When the data meets the expectations of data consumers, that data can be considered as high quality of data. The source of the data is frequently times the significant factor. Data entry or data transcription is inherently prone to bias or systematic errors both simple and complex. Data quality is one of the most critical issues in data management since dirty data frequently leads to inaccurate data analytic results and incorrect business decisions. The quality of data can be defined by two related factors [24]. They are firstly how well the data meets the expectations of data consumers and secondly how well the data represents the objects, events and concepts. In practice, data quality could also be a priority for specialists included with a large range of information systems, starting from the data warehousing and business intelligence to customer relationship management and supply chain management.

Particularly measuring the quality of data such as the use of data evolves and the amount of data is one of the most important challenges for data quality experts. This section briefly covers the most issues of data quality with a specific effort of duplicates. First, the idea of data quality dimensions as a method through which data quality will be measured and then completely different perspectives of data cleaning, which are usually performed before duplicate detection is covered. These broadly cited dimensions of data quality are as follows:

- Accuracy: Is that the data accurately representing the real-world entity or event?

- Consistency: Is that the data not containing syntactical anomalies and contradictions?
- Integrity: Are the relationships between attributes and entities consistent?
- Timeliness: Is that the data representing the real circumstance and is it available at the time needed?
- Completeness: Is all necessary data that are representing the entity or event present?
- Validity: Are all data values within the value domains specified?
- Uniqueness: Is there a single view of the data?

Quality data does not essentially mean the perfect data. But both data and schema dimensions are important. The low quality data can deeply influence the standard processes of business, whereas a schema of low quality. The major data quality problems can roughly be distinguished between single-source and multi-source problems and also between scheme-related and instance-related problems. They can be solved by data cleaning and data transformation. The problems of instance-level refer to data entry errors such as misspellings, redundancies or duplicates, contradictory values and inconsistencies in the actual data contents which are visible at the scheme level. These instance-level problems are the main effort of data cleaning. Thus, it focuses on the instance-level problems to be utilized on publication datasets.

2.2 An Overview of Data Cleaning

Organizations are obtaining huge amounts of data from different data sources in order to build the huge data repositories that control applications with the objectives of investing and more knowledgeable analytics. Data collection and acquisition regularly introduce errors in data such as missing values, typographical errors, improperly formatted entries, duplicated entries for the same real-world entity and violations of business and data integrity rules.

Data cleaning is the important process of data preparation for analyzing data in the analytical process by removing or modifying data that is incorrect, incomplete, inappropriate, repeated or improperly formatted in an integrating data warehouse or database. Data cleaning is not simply used to remove information but to make more space for new data and to increase the accuracy of datasets. Data cleaning includes other

activities such as fixing syntax and spelling errors, formatting data sets, fixing missing codes, empty fields and identifying duplicate data.

Data cleaning problems [22] typically consist of dealing with the lack of standardization in representing attributes, incomplete and missing data, determining usability, erroneous data, etc., Manual entry can also lead to incomplete, missing data and non-standard entry like naming conventions "Marry J." and "M.Jomes". Additional commonly problem in data cleaning can be the entry of duplicate data.

A survey mostly in data science and machine learning (ML) reveals that dirty data is the most common obstacle that has been faced dealing with data. With the popularity of data science, it has become progressively evident that data creation, unification, preparation and cleaning are key enablers in releasing the value of data. The development of efficient and effective data cleaning solutions is challenging and overflowing with deep theoretical and engineering issues. In any case of the type of data errors to be fixed, data cleaning activities usually contain two phases:

- Error detection where different errors are identified and probably validated by experts.
- Error repair where updates to the database are applied (or suggested to human experts) to pass the data to a cleaner state appropriate for downstream applications and analytics.

2.3 Record Matching Problem

With huge totals of data that are stored in data warehouses, mining information, and knowledge in databases has become a significant problem in recent researches. Data mining is the KDD process or the "knowledge discovery in databases" process as the analysis step.

A number of developing mining applications in information providing services such as data warehousing and online services need to combine information from different data sources to get better user performance, to improve the provided services, and to increase the business chances in organizations. These heterogeneous sources can be relational databases or web pages which provide information about the same real-world entities but describe these entities differently. It can be more storage space, long data retrieving time and wrong decision making by describing the same real-world object as different objects. The solving inconsistencies and different descriptions in

entities is the issue addressed in these different record values which describe the same semantic entity.

Detecting the possible duplicate records in a single database or multiple related databases is one of the abilities to do record matching. The duplicate detection is the main issue in Merge/Purge task which is to identify entry errors and combine multiple records. This task is also called data cleaning or data scrubbing. One significant research area of approximate record matching is the approximate string matching. Two different problems are considered in the survey by [6]. First, the description of equivalence allows only small differences within the two strings. The equivalence of two strings is the same as the mathematical concept of equivalence. Second, the similarity problem allows for more typing errors such as transposed letters, missing letters, etc. String matching has been one of the most considered issues in computer science. The best approach is based on edit distance.

2.4 Field Matching Techniques

Field Matching Technique is the inner stage of duplicate detection while the outer stage of duplicate detection is applied as the record matching technique. The duplicate detection depends on the string comparison techniques for resolving typographic variation in the string data and for errors in the numeric data. Resolving typographical errors can be critical in a record linkage. In case the comparisons of string pairs are done only in an exact character-by-character way, numerous matches may be lost.

The list of different techniques for field matching in the context of duplicate record detection includes:

- Character-based similarity measurement
- Token-based similarity measurement
- Similarity measurement of pronunciation
- Numerical similarity measurement

This section describes techniques that have been applied for matching fields with string data in the context of duplicate record detection. Character-based similarity metrics handle typographical errors well. In this proposed system, Edit Distance (or) Levenshtein Distance Algorithm is used to calculate field matching similarity scores. It covers the following similarity metrics:

Edit distance: Edit Distance is the minimum number of edit operations on a single character that is required to convert the string one into string two. Three types of edit operations are possible. They are:

- Insertion: insert a character into the string.
- Deletion: delete a character from the string.
- Substitution: replace a character with another character.

In the simplest procedure, one edit operation has cost 1. This kind of edit distance is also mentioned as Levenshtein distance [7]. Needleman and Wunsch [8] improved the original edit distance model and allowed for different costs for different edit distance operations. The edit distance metrics are more suitable for detecting typographical errors but they are typically ineffective for other types of mismatches.

Affine gap distance: The edit distance metric does not effort well when matching strings that have been shortened (e.g, Hana R. Smith. vs. Hannar Richard Smith.). The affine gap distance metric [9] offers a solution to this problem by introducing two extra edit operations: open gap and extend gap. The extending gap cost is usually smaller than the opening gap cost and this result in smaller cost drawbacks for gap mismatches than the equivalent cost under the edit distance metric.

Smith-Waterman distance: Smith and Waterman [10] defined an extension of edit distance and affine gap distance, in which mismatches at the beginning and the end of strings have lower costs than mismatches in the middle. This metric lets for better strings local alignment (i.e., substring matching). So, the two strings "Prof. Mary R. Jones, University of Calgary" and "Mary R. Jones, Prof." can match in a short distance by using the Smith-Waterman distance because the prefixes and suffixes are ignored. The distance between two strings can be calculated using a dynamic programming technique to find the lowest cost of changes that converts one string into another. Pinheiro and Sun [11] suggested a similar similarity measure which tries to find the best character alignment for the two compared strings s1 and s2 so that the number of character mismatches is minimized.

Jaro distance metric: Jaro [12] presented a string comparison algorithm that was fundamentally used for comparison of first and last names. The calculating the Jaro metric algorithm for two strings s1 and s2 includes the following stages:

- Compute the string lengths |s1| and |s2|.
- Search the *"common characters" c* within the two strings.
- Search the "*number of transpositions" t* that is the number of transpositions. It is calculated as follows by comparing the i^{th} common character in s1 with the i^{th} common character in s2. Each non-matching character is a transposition.

Q-gram distance: The q-grams are short character substrings of length q of the database strings [13]. The intuition behind the use of q-grams as a basis for approximate string matching is that when two strings s1 and s2 are similar, they share a large number of q-grams in common. Given a string s, its q-grams are gained by sliding a window of length q over the characters of a string s. Since q-grams at the starting and the end of the string can have less than q characters from s, the strings are theoretically expanded by "padding" the starting and the end of the string with q - 1 occurrences of a special padding character, not in the original alphabet.

2.5 Record Matching Techniques

The record consists of multiple fields and it can make the problem of duplicate detection rather more complex. The field matching and string matching methods will be applied to match individual fields of a record. There also are various record matching approaches to solve the record matching problems. This section reviews various approaches during this category mostly knowledge-based, distance-based, and induction-based addition as supervised learning and unsupervised learning.

2.5.1 Notation

The two tables '*A*' and '*B*' have '*n*' comparable fields. Assume that these two tables are wanted to match without loss of generality. Within the record matching problem, each record pair (α, β); $(\alpha \in A, \beta \in B)$ is allocated to at least one of the two classes '*M*' and '*N*'. The category '*M*' contains the "match" record pairs that represent the same entity and also the class '*N*' contains the "non-match" record pairs that represent two different entities.

Each record pair (α, β) is presented as a random $vector\ x = [x_1, \ldots, x_n]^T$, where T denotes the transpose of the vector with n components that relate to the n comparable fields of tables A and B. Each x_i shows the level of agreement of the i^{th} field for the records α and β. Several approaches use binary values for the x_i's. Set $x_i = 1$ if field i agrees and $x_i = 0$ if field i disagrees.

2.5.2 Probabilistic Matching Models

Newcombe et al. [14] was the principal to acknowledge the duplicate detection Bayesian inference problem. Then, Fellegi and Sunter [15] formalized the intuition of Newcombe et al. and introduced the notation which is additionally employed in duplicate detection literature. The comparison $vector\ x$ is the input to a decision rule that assigns x to M or to N. The most assumption is that x is a random vector whose density function is different for each of these two classes. Then, if the density function for every class is known, the duplicate detection problem becomes a Bayesian inference problem.

2.5.3 Supervised and Semi-Supervised Learning

A Bayesian approach is used in the probabilistic model to classify the record pairs into two classes, M and N. This model was widely used in duplicate detection tasks as an application of the Fellegi-Sunter model. While the Fellegi-Sunter approach dominated the field for quite twenty years, the development of new classification techniques in the machine learning and statistics communities encouraged the development of new deduplication techniques. The supervised learning systems depend upon the existence of training data within the variety of record pairs, pre-labeled as matching or not.

2.5.4 Active-Learning-Based Techniques

One of the problems with the supervised learning techniques is the need for a huge number of training examples. Whereas it is simple to make a huge number of training pairs that are either obviously non-duplicates or exactly duplicates, it is very difficult to produce the ambiguous cases that would help to form a highly precise classifier. Based on this observation, some duplicate detection systems used active learning techniques [16] to automatically locate such ambiguous pairs. Not at all like

an "ordinary" learner that is trained using a static training set, an "active" learner effectively choices subsets of instances from unlabeled data, which, when labeled, will provide the highest information gain to the learner.

2.5.5 Distance-Based Techniques

The way of avoiding the need for training data or some human effort to create the matching models is to define a distance metric for records which does not need modification through training data. It is probable to match similar records without the requirement for training using the distance metric and an appropriate matching threshold. One approach is to treat a record as a long field and use one of the distance metrics described in Section 2.4 to decide which records are similar or not. Monge and Elkan [17] suggested a string matching algorithm for detecting extremely similar records. The basic idea is the applying a general purpose field matching algorithm, especially one that is able to account for distance in the strings and to play the role of the duplicate detection algorithm. The distance-based approaches that conflate each record in one big field may ignore the important information that can be used for duplicate detection. A simple approach uses the appropriate distance metric for each field to measure the distance between individual fields and then calculates the weighted distance between the records.

2.5.6 Rule-Based Approaches

The rule-based approaches can be considered as distance-based techniques, where the distance of two records is either 0 or 1 by using the rules to define whether two records are the same or not. Yu Jiang and Can Lin [18] proposed a rule-based method for de-duplicating article records across databases. Table 2.1 shows as an example which an expert might define rules.

Table 2.1 Example of Defined Rules

IF age < 22	THEN status = undergraduate ELSE status =graduate
IF distanceFromHome > 10	THEN transportation = car ELSE transportation = bicycle

2.5.7 Unsupervised Learning

Ravikumar and Cohen [19] follow a similar approach and propose a graphical, hierarchical model for learning to match record pairs. The basis of this method is to model each field of the comparison vector as a latent binary variable which shows whether the two fields match or not. Bhattacharya and Getoor [20] proposed to use the Latent Dirichlet Allocation generative model to perform the duplicate detection. In this model, the latent variable is a unique identifier for each entity in the database.

2.6 Improving the Efficiency of Duplicate Detection

The process of identifying whether two records refer to the same real-world object, we have focused primarily on the quality of comparison procedures and not on the efficiency of the duplicate detection process. The fundamental issue of improving the duplicate detection speed is described in this section. Blocking and windowing methods can be applied to decrease the cost of record comparison in the efficiency of record comparison improvement. The most significant characteristic for windowing is the Sorted Neighborhood Method (SNM).

2.6.1 Sorted Neighborhood Approach

Using a sorted neighborhood approach can reduce the cost of comparing records and increase the efficiency of comparing records. Hernáandez and Stolfo [21] describe the sorted neighborhood approach. The sorted neighborhood approach involves three steps:

- Create sorting key: A key for each record in the dataset is allocated to each record. Keys are created by concatenating the values of two or more attributes.
- Sort the data: The records in the database are sorted based on the sorting key.
- Merge: A fixed size window is moved through the list of records sequentially to limit the comparison of records matching to those records in the window. Each new record that enters this window is compared to the previous record to find a “matching” record.

The Sorted Neighborhood Method uses fixed size windows. If the selection of window size is too small, some actual duplicate records may be lost and using larger window size will often result in unnecessary comparisons within the window. The

effectiveness of the sorted neighborhood approach depends greatly on the creation of sorting keys. Figure 2.1 shows an example of the sorted neighborhood method.

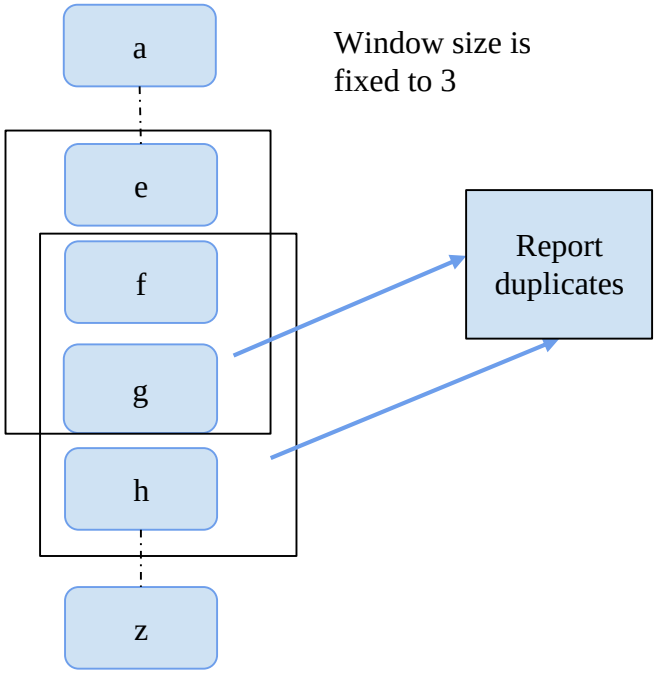


Figure 2.1 Sorted Neighborhood Method (SNM)

2.6.2 Duplicate Count Strategy Approach

The Duplicate Count strategy (DCS) is the extension of the Sorted Neighborhood Method (SNM). It is based on the windowing methods and varies the window size based on the number of detected duplicate pairs. The set of compared records differs from the original SNM because of the increase and decrease of the window size. Changing the window size does not certainly result in additional comparisons but it can also reduce the number of comparisons in the detection process. However, adapting the window size should result in a higher efficiency for a given effectiveness or in an overall higher effectiveness for a given efficiency.

DCS uses the number of records in the window as an initial window size. The more duplicates of a record are found within a window if the window is larger. On the other hand, if a duplicate of a record in its neighborhood is not found, at that point we expect that there are no duplicates or the duplicates are very far away within the sorting order. The record is compared with w - 1 successors in the beginning step. So the current window can be defined as $W\ (i, i\ +\ w\ -\ 1)$. If no duplicates can be found in this

window, there is no need to increase the window. But if there is at least one duplicate, then start increasing the window.

2.6.2.1 Basic Strategy

The basic strategy includes increasing the window size by one record. The basic duplicate count strategy involves the following steps:

1. Assign the sorting key to each record and sort the records.
2. Create the window with initial window size *w*.
3. Compare the first record with all other records in the window.
4. Increase the window size while $\frac{\text{detected duplicates}}{\text{comparisons}} \geq \phi$ (ϕ: average number of comparisons per duplicate)
5. Slide the window (initial window size *w*)
6. Calculate transitive closure.

   Figure 2.2 shows the overview of duplicate count strategy steps.

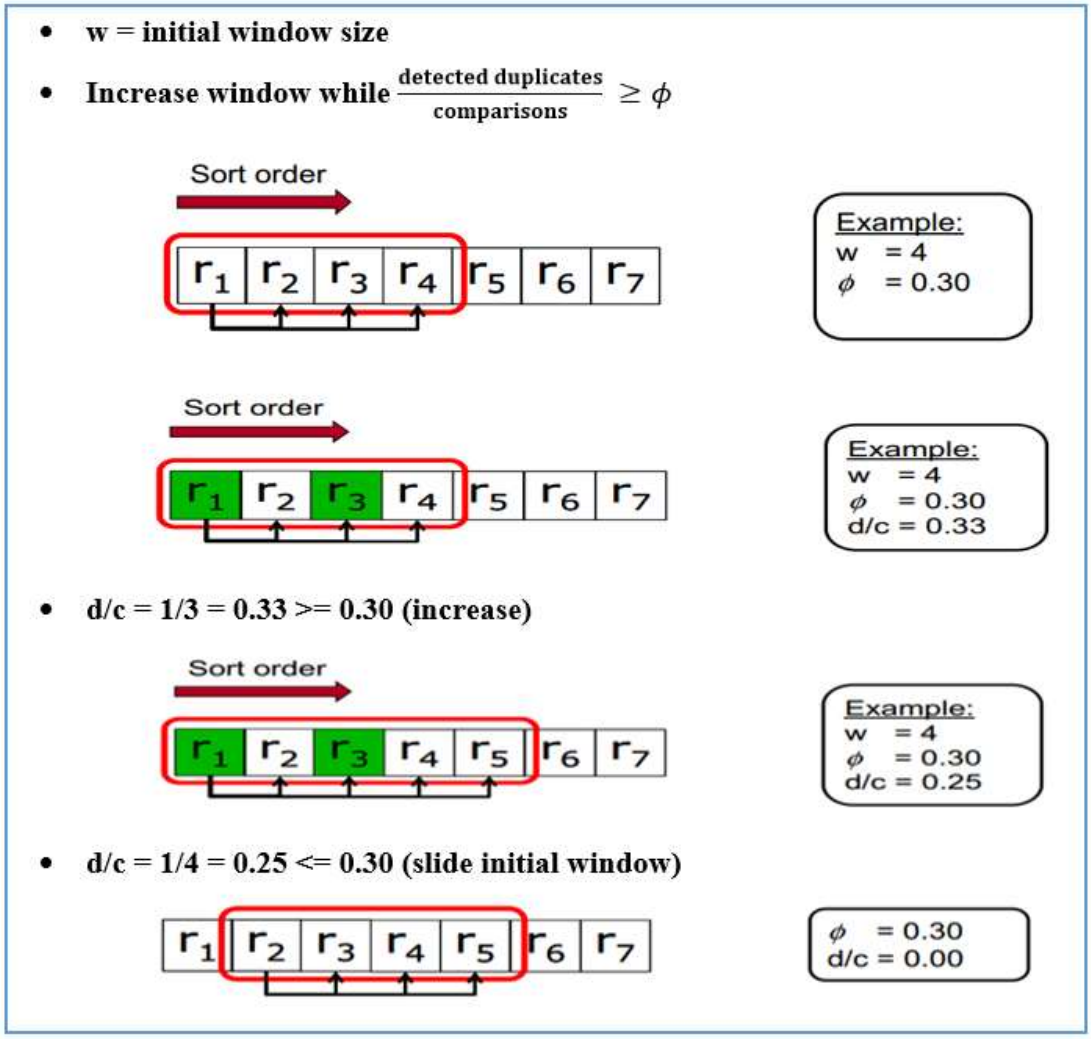


Figure 2.2 Overview of Duplicate Count Strategy (DCS)

2.6.2.2 Multi Record Increase Approach

DCS++ is an enhancement of the basic strategy by increasing the variant multiple records. There are two main ideas in the multi record increase approach instead of increasing the window by just one record. First, if each duplicate is found, the next *w-1* adjacent records of that duplicate are added to the window even if the average is lower than the threshold ϕ. Second, windows for duplicates have been omitted to save the comparisons. Skip window for r_3 in Figure 2.3. It uses the transitive closure to find additional duplicates and to save some of the record comparisons. Assume that the record pairs (r_1, r_2) and (r_1, r_3) are duplicates, with $1 < 2 < 3$. Calculating the transitive closure returns the additional duplicate pair (r_2, r_3). Therefore, there is no need to check the window $W(k, k + w - 1)$ and then this window is skipped. There is no loss in the window because the window for r_1 covers all comparisons that r_3 would have made.

Figure 2.3 shows the overview of duplicate count strategy multi record increase approach.

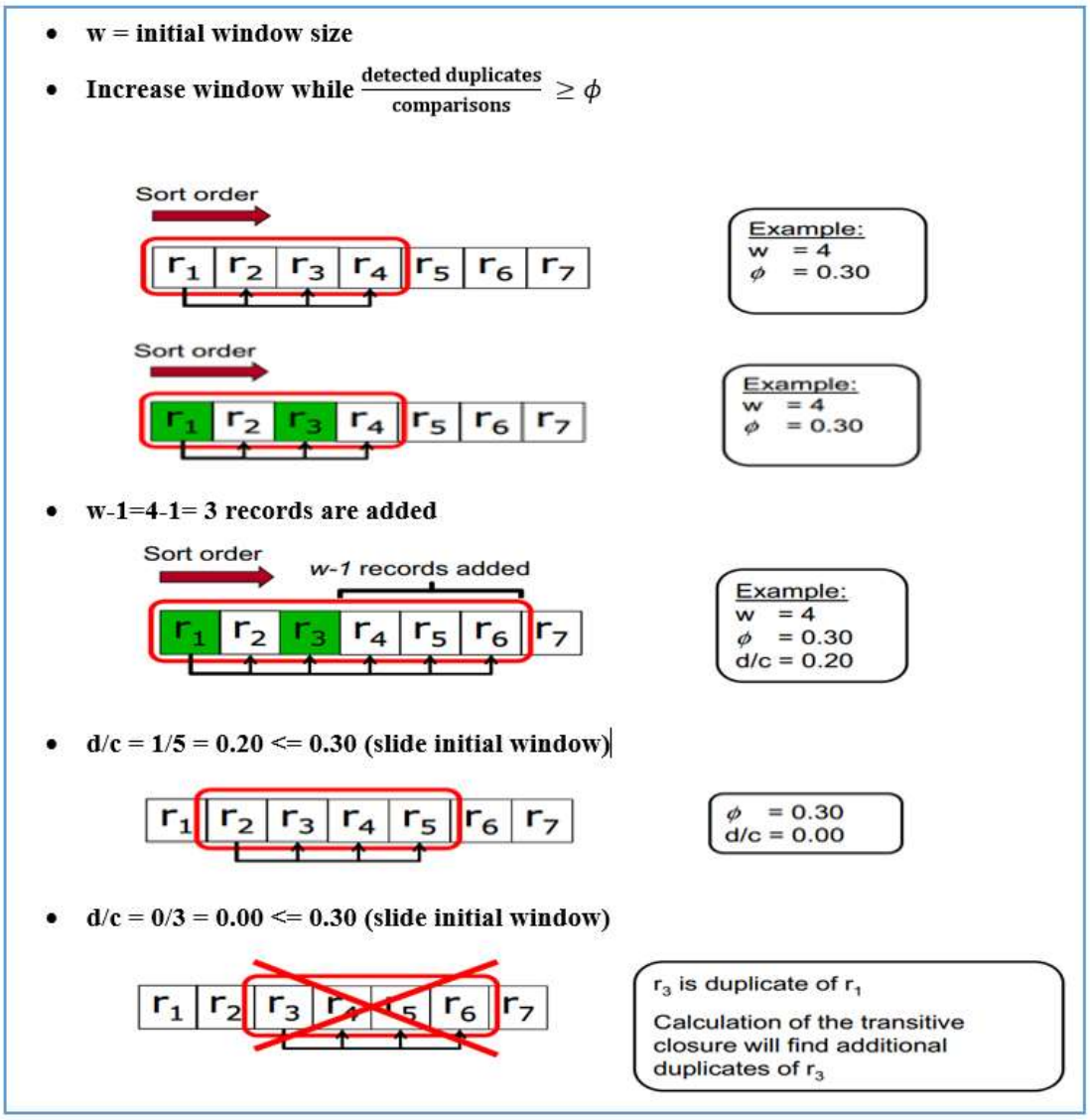


Figure 2.3 Overview of Duplicate Count Strategy-Multi Record Increase (DCS++)

CHAPTER 3

DESIGN OF THE PROPOSED SYSTEM

The main purpose of this system is to detect the possible duplicate record pairs in the dataset. The repeated data takes up a lot of storage space in the data warehouse and takes longer for retrieving the exact data. There are two main tasks for duplicate detection. They are the field matching technique as the inner stage of duplicate detection and the record matching technique as the outer stage of duplicate detection. In this system, Edit Distance (or) Levenshtein Distance Algorithm is used for field matching and DCS++ Algorithm is used to detect duplicate record pairs. For dataset, the Cora publication dataset is used as a case study for the experiment. The system is implemented on the Window and MacOS platforms with the PHP programming language and XAMPP cross-platform web server. PHP is an acronym for Hypertext Preprocessor (earlier called, Personal Home Page) [25].

3.1 Overview of the Proposed System

Figure 3.1 shows the overview of the proposed duplicate detection system. Data preparation is performed by parsing from the input xml raw dataset and standardizing parsed data which can lead to fast identification of duplicates. After the data preparation phase, the data are normally stored in such a way to be easily compared in next phases. DCS++ is one of the improvements of SNM (Sorted Neighborhood Method). So, key creation and sorting are the same phases of SNM but DCS++ do not use the fixed window size. It is based on the windowing method. The records are sorted based on the sorting key to compare the possible duplicates by keeping the same record next and then slides a window of adaptive size sequentially over the sorted records. All records within such a window are compared with each other and identified as candidate duplicates.

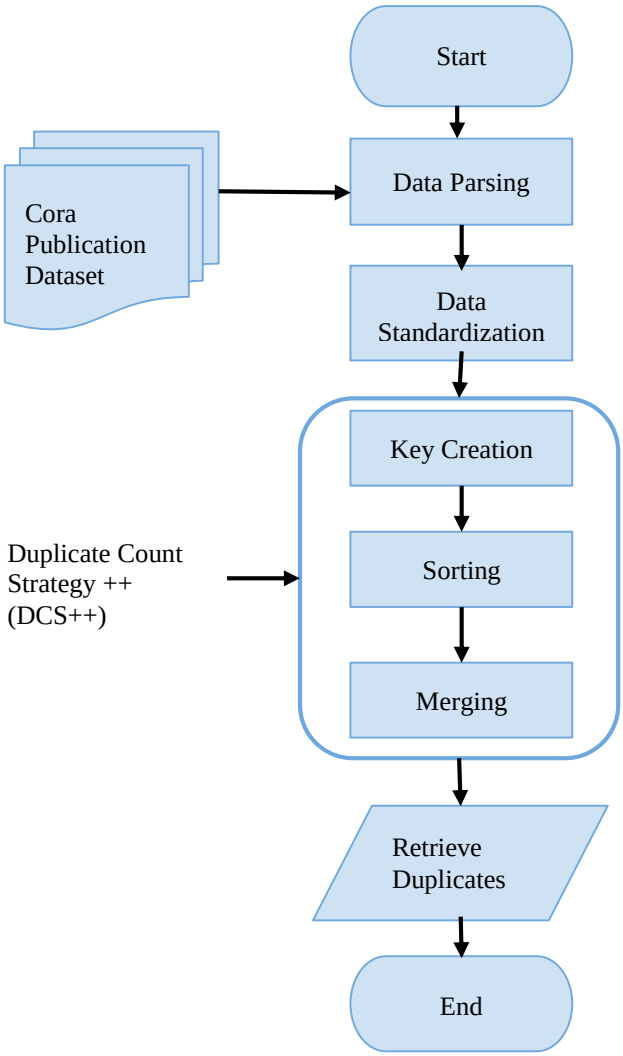


Figure 3.1 Overview of the Proposed System

3.2 Overview of Cora Publication XML Dataset

In this system, XML Cora Dataset is used as an experimental dataset for input records and parsing them to detect syntax errors. This dataset contains bibliographic information for scientific papers. It provides 1,879 objects.

The Cora dataset is prepared by the original Andrew McCallum and his versions of this dataset are provided on his data web page [26]. Many publications in record linkage and entity records over the years used these various versions of the Cora dataset. Figure 3.2 shows the example of publication XML Cora dataset.

```
<CORA>
<NEWREFERENCE id="1">
ahlskog1994a
<author>
M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O.</author>
<title>Inganas and M.R.</title>
<journal>Andersson, J Appl. Phys.,</journal>
<volume>76,</volume>
<pages>893,</pages>
<date>(1994).</date>
</NEWREFERENCE>
<NEWREFERENCE id="2">
ahlskog1994a
<author> M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O.
Inganas and M.R.   Andersson, </author>
<journal>  J Appl. Phys., </journal>
<volume> 76, </volume>
<pages>893, </pages>
<date> (1994). </date>
</NEWREFERENCE>

…
<NEWREFERENCE id="9">
asfahl1992a
<author> C. RayAsfahl. </author>
<title> Robots and Manufacturing Automation. </title>
<publisher> John Wiley and Sons, </publisher>
<address> New York, </address>
<note> second edition, </note>
<date> 1992. </date>
</NEWREFERENCE>
<NEWREFERENCE id="10">
benford1993a
<author> Steve Benford and Lennart E. Fahlen. </author>
<title> A spatial model of interaction in large virtual environments. </title>
<booktitle> In Proceedings of ECSCW'93, </booktitle>
<address> Milan, </address>
<date> 1993. </date>
</NEWREFERENCE>

      ...
</CORA>
```

Figure 3.2 Sample XML Cora Dataset

3.3 Data Preprocessing

The duplicate record detection needs a data preprocessing phase that is a necessary step in data cleanup before the process of duplicate detection. The data preprocessing phase involves data parsing, data transformation and standardization procedures. The data preparation techniques are also described in terms of ETL (extraction, transformation, loading) [24].

In this system, parsing and data standardization of preprocessing phase must be performed first to increase the quality of in-flow data and the second to make the data comparable and more usable.

3.3.1 Data Parsing

Data parsing is the first main component in the data preprocessing phase of the matching record. The data field is easier to correct, standardize and match data by parsing it because the data parsing allows comparing the individual components rather than long strings of data.

In this system, Cora publication XML dataset is used as input. And then it includes the removing xml tags in publication records and parsing them to detect syntax errors. Example of a parsed XML dataset is shown in Table 3.1.

Table 3.1 Example of Parsed Cora Publication XML Dataset

ID	REF Name	Author	Title	Date	...
R1	ahlskog1994a	M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O.	Inganas and M.R.	(1994).	
R2	asfahl1992a	C. RayAsfahl.	Robots and Manufacturing Automation.	1992.	
R3	benford1993a	Steve Benford and Lennart E. Fahlen.	A spatial model of interaction in large virtual environments.	1993.	
R4	benford1994a	Benford, S., and Fahln, L.	Viewpoints, Actionpoints and Spatial Frames for Collaborative User Interfaces,	June 1994,	
R5	carlson1993a	Carlson, C. ; L.E.Fahln.	Integrated CSCW Tools Within a Shared 3D Virtual Environment.	1993.	

3.3.2 Data Standardization

Data standardization is the process of standardizing the represented information in some fields to a uniform specific format. There may be different data formats in different records which come from a variety of data sources and need to be converted to a uniform representation prior to the process of detecting duplicates. If there is no data standardization, numerous duplicate entries might be chosen as non-duplicates wrongly. It is based on the fact that common identifying information cannot be compared.

In this system, author name, date and title are standardized. Author names can be all authors participating in the publication. But only the first author is extracted and formatted into the first character of First Name, dot (.) and Last Name only. Date includes one or combination of year, month and dates. The system extracts only the year from date value and title must not be empty. Therefore, these preprocessed data fields can be easily used in the key creation and possible duplicate records detection processes. Table 3.2 shows an example of standardized fields in record.

Table 3.2 Example of Standardized Fields

Author Names	Standardized Author Names		Date	Standardized Date
	First Name	Last Name		
Steve Benford and Lennart E. Fahlen.	S.	Benford	1993.	1993
Brown, D. F., Moura, H. and Watt, D. A.	D.	Brown	(1992b),	1992
B. Buth et. al.,	B.	Buth	1992,	1992
Benford, S., Bowers, J., Fahln, L., Greenhalgh, C., and Snowdon, D.,	S.	Benford	May 7-11, 1995,	1995
Daelemans, W., Van den Bosch, A., and Weijters, T.	W.	Daelemans	(1989).	1989

3.4 Duplicate Detection

The system has mainly focused on empirical algorithms which contain sorting, windowing and blocking methods. In this section, possible duplicate records are detected by using duplicate count strategy-multi record increase approach (DCS++). It is an enhancement of the sorted neighborhood method and basic duplicate count strategy (DCS). It tends to decrease the number of record comparisons when records in a dataset are detected as duplicate. Firstly, the key creation process described in section 3.4.1 must be performed to sort the records in a dataset and to detect near duplicates easily. And then, these sorted records are used as inputs to DCS++ algorithm for detecting duplicates. These record pairs are identified by DCS++ algorithm and then computed their similarity scores using Levenshtein distance algorithm.

3.4.1 Key Creation

During the key creation process, a sorted key is generated for each record in the dataset by extracting the relevant fields or portions of fields from a significant discriminating attribute. The effectiveness of DCS++ algorithm highly depends on the selection of keys in key creation process to sort the records. The process of key selection is a highly knowledge-intensive and domain specific process [23].

In this system, a key consists of the combination of the first letter of First Name, three consonants of Last Name, last two digits of Date field and four consonants of Title field which are included in preprocessed records. These choices are made since the domain expert determined that last names are usually misspelled due to errors in vowels, vocalized sounds. Table 3.3 shows an example of records and keys used in this system.

Table 3.3 Example of Key Creation

RID	First	Last	Date	Title	Key
R1	**C**.	**R**a**y**As**f**ahl	19**92**	**R**o**b**o**ts** and Manufacturing Automation.	**CRYS92RBTS**
R2	**S**.	**B**e**nf**ord	19**93**	A **sp**a**t**i**al** model of interaction in large virtual environments.	**SBNF93SPTL**
R3	**D**.	**Br**o**w**n	19**92**	A**ctre**ss: an action semantics directed compiler generator,	**DBRW92CTRS**
R4	**B**.	**B**u**th**	19**92**	**Pr**o**v**a**b**ly correct compiler	**BBTH92PRVB**

				development and implementation.	
R5	**C**.	**C**a**rl**son	1993	Integrated CSCW Tools Within a Shared 3D Virtual Environment.	CCRL93NTGR

3.4.2 Sorting Phase

The records are sorted in the data list based on the foundation of the key selected in the earlier phase. A sorting key is characterized to be a sequence of attributes or a sequence of substrings inside the attributes chosen from the preprocessed record in an important manner. The sorted keys are used for sorting the entire dataset with the purpose that all matching or possible duplicate records will appear close to each other in the final sorted list for this system.

3.4.3 Merging Phase

After preprocessed Cora publication XML records have been sorted in sorting phase, the system uses DCS++ algorithm through the consecutive list of records limiting the comparisons for duplicate detection. DCS++ is an improvement of sorted neighborhood method by adapting the window size to detect duplicates effectively and by removing the skipped window to reduce the number of comparisons.

3.4.3.1 Duplicate Count Strategy++ Algorithm

In this experiment, the initial window size (w) is provided as 20 and DCS++ threshold (Ø) is recommended as $\frac{1}{w-1}$ not to miss any duplicates. The detail of Duplicate Count Strategy-Multi Record Increase (DCS++) algorithm is described in Figure 3.3.

Algorithm: Duplicate Count Strategy-Multi Record Increase Algorithm

Require: w > 1 and 0 < Ø ≤ 1 (w: initial window size, Ø: DCS++ threshold)

1. sort records by sorting key
2. populate window win with first w records of records
3. skipRecords ← null /* records to be skipped */

   /* iterate over all records and search for duplicates */

```
4.  for  j = 1 to  records.length - 1  do
5.    if win[1]  NOT IN  skipRecords  then
6.      numDuplicates ← 0              /* number of detected duplicates */
7.      numComparisons ← 0             /* number of comparisons */
8.      k ← 2
        /* iterate over win to find dup. of rec win[1] */
9.      while k  ≤  win.length do
          /* check if record pair is a duplicate */
10.       if   isDuplicate (win[1] , win[k] ) then
11.             emit duplicate pair (win[1] , win[k])
12.             skipRecords.add ( win[k] )
13.             numDuplicates ← numDuplicates + 1
                /* increase window size from k by w-1 records */
14.             while win.length < k+w-1 and j + win.length < records.length do
15.                   win.add (records [ j + win.length + 1] )
16.             end while
17.       end if
18.       numComparisons ← numComparisons+1
          /* potentially increase window size by 1 */
19.       if k = win.length  and j + k < records.length and (numDuplicates /
          numComparisons )  ≥ Ø then
20.             win.add (records [j + k - 1])
21.       end if
22.       k  ← k + 1
23.     end while
24.   end if
      /* slide window */
25.   win.remove(1)
26.   if win.length < w   and   j + k  <  records.length then
27.     win.add (records [j + k - 1])
28.   else       /* trim window to size w */
29.     while win.length  >  w  do
```

```
30.        win.remove (win.length)     /* remove last record from win */
31.      end while
32.    end if
33.    j ← j  +  1
34. end for
35. calculate transitive closure.
```

Figure 3.3 Duplicate Count Strategy-Multi Record Increase Algorithm

3.4.3.2 Matching Criteria

This system uses the publication XML Cora dataset, which stores bibliographic information in various fields of scientific publications papers. And comparisons between the records within that dataset are performed according to four matching criteria. That is, similarity between these criteria is computed using dynamic programming algorithms.

The most data fields in a Cora publication record are the free-text strings. To do string comparison, this system uses an edit distance dynamic programming algorithm. And there are four matching criteria for the system:

- E(Key) = String edit distance of key field
- E(Title) = String edit distance of title field
- E(Author) = String edit distance of author field
- E(Date) = String edit distance of date field

3.4.3.3 Similarity Measuring

Similarity measuring must be implemented for field matching and string matching. There are various algorithms to use such as Edit Distance, N-grams algorithm, Smith Waterman algorithm, Jaro algorithm and Text Similarity Measure algorithm. Among them, this system uses the Edit Distance (or) Levenshtein Distance algorithm as it is a widely used metric to define the string similarity.

Levenshtein Distance Algorithm: Levenshtein distance is a metric for measuring the amount of difference between two sequences. The Levenshtein distance between two strings is defined as the minimum number of edits necessary to transform

one string into the other string with the allowable edit operations such as insertion, deletion and substitution of a single character. Typographical errors stem from mistyping ("hte" vs. "the") whereas cognitive errors are actual mistakes. It is also related to note that spelling errors can be single or numerous error misspellings. In this system, Levenshtein Distance Algorithm is used for checking the words.

For example, the Levenshtein distance between "survey" and "surgery" is 2, since these two edits change one string into the other and there is no way to do it with fewer than two edits:

d (v,w) = minimum number of elementary operations to transform v => w.

For example,	v = survey	survey	
	w = surgery	surgey	substitute (+1)
		surgery	insert (+1)

Levenshtein distance d (v,w) = 2

Similarity Score = 1 - (2 / 7) = 0.71

Let P be $(|v| + 1) \times (|w| + 1)$ matrix where $P_{i,j}$ represents the minimum number of operations to match $v_1, v_2, \dots, v_i$ with $w_1, w_2, \dots, w_j$. The matrix P is constructed as follows:

$$\boldsymbol{P_{i,0} = i}$$

$$\boldsymbol{P_{0,j} = j}$$

$$\boldsymbol{P_{i,j} = If\ v_i == w_j\ , P_{i-1,j-1}}$$

$$\boldsymbol{Otherwise\ 1 + min(P_{i-1,j}, P_{i,j-1}, P_{i-1,j-1})}$$

Table 3.4 shows the example of distance matrix using Levenshetin's Algorithm and Figure 3.4 shows the Levenshtein Distance Algorithm.

Table 3.4 Sample Distance Matrix using Levenshtein's Algorithm

		S	**U**	**R**	**V**	**E**	**Y**
	0	1	2	3	4	5	6
S	1	0	1	2	3	4	5
U	2	1	0	1	2	3	4
R	3	2	2	0	1	2	3
G	4	3	3	1	1	2	3
E	5	4	4	2	2	1	2
R	6	5	5	4	3	2	2
Y	7	6	6	5	4	3	**2**

```
int LevDistance (Strl, Str2)
{
        for i from 0 to lenStr1
                d[i, 0] := i
        for j from 0 to lenStr2
                d[0, j] := j
        for i from 1 to lenStr1
                for j from 1 to lenStr2
                        if str1 [i] = = str2 [j]   then cost := 0
                                                   else cost := 1
                        d[i, j] := minimum         (
                                                     d[i-1, j] + 1,       // deletion
                                                     d[i, j-1] + 1,       // insertion
                                                     d[i-1, j-1] + cost // substitution
                                                   )
        return  d[lenStrl, lenStr2]
}
```

Figure 3.4 Levenshtein's Distance Algorithm

Field Similarity: Let X and Y be records and $f_{x1}, f_{x2}, \ldots, f_{xn}$ be the tokens of the corresponding fields in record X. The tokens of the fields in record Y be $f_{y1}, f_{y2}, \ldots, f_{ym}$. For calculating the field similarity, each token f_{xi} , $1 \leq i \leq n$ is compared with tokens f_{yi}, $1 \leq i \leq m$. The field similarity for X and Y:

$$Sim_F(X,Y) = \left(\sum_{i=1}^{n} f_{xi} + \sum_{i=1}^{m} f_{yi}\right) / (n+m) \qquad \textbf{(3.1)}$$

Record Similarity: Assume a database has fields $F_1, F_2, F_3, \ldots, F_n$ with field weightages $W_1, W_2, W_3, \ldots, W_n$ respectively. The record similarity for X and Y:

$$\sum_{i=1}^{n} Sim_F(X,Y) \times W_i \qquad \textbf{(3.2)}$$

In the proposed system, Equation 3.1 is used to compute the similarity values of Key, Title, Author and Date between two fields. Equation 3.2 is used to compute the similarity value of records. The field weightages are assigned as 0.2 in Key, 0.5 in Title,

0.2 in Author and 0.1 in Date. Our field similarity threshold except date field and record similarity threshold such as 0.7 are duplicate records by using Equation 3.2 and therefore, it should be merged. If the similarity score is equal to one, the two records are a perfect match. If a record pair has a similarity value that is equal or higher than the similarity threshold, it is considered as duplicate records, otherwise non-duplicate.

Table 3.5 and Table 3.6 show the sample calculations of similarity score between records by using Equation 3.1 and 3.2 with threshold 0.7. In Table 3.5, the resulting similarity score between R1 and R2 is less than the similarity threshold. Therefore, the system assumed that R1 and R2 are non-duplicated records.

Table 3.5 Sample Calculation of Similarity Score in Two Records (R1, R2) with Threshold 0.7

	Author	**Date**	**Title**	**Key**
R1	C.RayAsfahl	1992	Robots and Manufacturing Automation.	CRYS92RBTS
R2	S.Benford	1993	A spatial model of interaction in large virtual environments.	SBNF94VWPN
Levenshtein Distance	9	1	44	9
$\boldsymbol{Sim_F(R1, R2)}$	0.19 (19%)	0.75 (75%)	0.28 (28%)	0.1(10%)
$\boldsymbol{Sim_F(R1, R2) \times W_i}$	0.19 * 0.2 = 0.038	0.75 * 0.1 = 0.075	0.28 * 0.5 = 0.14	0.1 * 0.2 = 0.02
$\sum_{i=1}^{n} \boldsymbol{Sim_F(R1, R2) \times W_i}$ = 0.038 + 0.075 + 0.14 + 0.02 = **0.273** **(result: Non-duplicate pair)**				

In Table 3.6, the resulting similarity score is equal to the similarity threshold 0.7. Therefore, R3 and R4 are the duplicated records.

Table 3.6 Sample Calculation of Similarity Score in Two Records (R3, R4) with Threshold 0.7

	Author	**Date**	**Title**	**Key**
R3	A.Bruce	1994	Goal-directed Classification Using Linear Machine Decision Trees.	ABRC94GLDR
R4	C.Brodley	1994	Goal-directed Classification Using Linear Machine Decision Trees.	CBRD94GLDR
Levenshtein Distance	5	0	0	2
$\boldsymbol{Sim_F(R3, R4)}$	0.45(45%)	1(100%)	1(100%)	0.02 (80%)
$\boldsymbol{Sim_F(R3, R4)} \times \boldsymbol{W_i}$	0.45 * 0.2 = 0.09	1 * 0.1 = 0.1	1 * 0.5 = 0.5	0.02 * 0.2 = 0.004
$\sum_{i=1}^{n} \boldsymbol{Sim_F(R3, R4)} \times \boldsymbol{W_i}$ = 0.09+0.1+0.5+0.004 = **0.694** **(result: Duplicate pair)**				

Table 3.7 shows the summary of calculations in Table 3.5 and Table 3.6 for sample records such as R1, R2, R3 and R4. Table 3.7 displays the similarity score of each field between two records and then the system assumes that the records in a comparison as a duplicate record or not by using Equation 3.2 for calculation of record similarity.

Table3.7 Example of Field Similarities and Record Similarities

RID	RID	Author Sim	Date Sim	Title Sim	Key Sim	$\sum_{i=1}^{n} Sim_F(X,Y) \times W_i$	Result
R1	R2	0.038	0.075	0.14	0.02	0.3	Non duplicate pair
R3	R4	0.09	0.1	0.5	0.004	0.7	Duplicate pair

3.5 Performance Evaluation

In this system, the performance of the algorithm is measured using: **Recall**, **False Positive Error** (FP), **False Negative Error** (FN) and **Precision**.

Recall

The percentage of duplicate records is that the system correctly identifies. Recall percentage is computed by following equation:

$$\textbf{Recall} = \frac{no\ of\ identified\ duplicates}{no\ of\ actual\ duplicates} \times 100\% \qquad \textbf{(3.3)}$$

False Positive Error (FP)

The percentage of records incorrectly identified as duplicates. FP percentage is defined as the equation:

$$\textbf{FP} = \frac{no\ of\ wrongly\ identified\ duplicates}{total\ number\ of\ identified\ duplicates} \times 100\% \qquad \textbf{(3.4)}$$

False Negative Error (FN)

The percentage of duplicate records is that the system does not detect. FN percentage is computed by following equation:

$$\textbf{FN} = 100\% - \textbf{Recall} \qquad \textbf{(3.5)}$$

Precision

The percentage of information reported as relevant by the system that is correct. Precision percentage is defined as the equation:

$$\textbf{Precision} = 100\% - \textbf{FP} \qquad \textbf{(3.6)}$$

CHAPTER 4

IMPLEMENTATION OF THE PROPOSED SYSTEM

In this system, the possible duplicate records in XML Cora publication dataset are detected by using the DCS++ approach and Levenshtein distance algorithm. The data preparation step must be done before duplicate detection. And then, these preprocessed records become the input to duplicate detection process. According to the proposed approach, key creation process must be performed firstly to sort the records in dataset. These sorted records are used in field matching and record matching to detect as duplicate pairs.

This chapter describes the software and hardware requirements for environmental setup before running the system, the program interface designs for each step of data preparation process such as data parsing, data standardization and proposed duplicate records detection process. After that, based on the results of the duplicate detection process, performance evaluation of the system is described.

4.1 Experimental Setup

In order to evaluate the proposed algorithms, install XAMPP [27] on Windows or OS X. Apache service is started in the XAMPP Control Panel as a local server. XAMPP is the most popular in web development environment. XAMPP is a totally free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The open source package of XAMPP has been set up to be extremely easy to install and to use. The experimental environments are as follows:

- Hardware configurations
    - Operating System: Windows 8, macOS Catalina
    - CPU: Core i5@3 GHz
    - Memory: 8 GB

- Software requirements
    - XAMPP version: 7.3.0 64 bit, and
    - PHP version: 7.3.0

4.2 Implementation of the System

When the system is started, it appears as shown in Figure 4.1. It shows the buttons that are representing each step of the duplicate record detection process and the process of changing window size and similarity threshold for getting different results. The About menu section shows an overview of the proposed system and algorithms that are used in this system. And, the last menu of the system, Contact menu, allows us to view the information of our university.

Figure 4.1 shows the Home Page, there are five buttons to view each process of duplicate record detection in the proposed system. Raw Dataset button shows the parsed Cora data records in a table form, Standardized Dataset button shows the standardized data fields, Extract Keys button presents the keys in key creation, Sorting Keys button represent the process of sorting phase before applying the duplicate detection and Duplicate Detection button shows the final getting result of detection process on Cora dataset with preset window size and similarity threshold value.

In the changing window size and threshold section, the user can select the provided window size and the similarity threshold value for testing different detecting results instead of using default values. The domain experts said that generally the window size is mostly used between 10 and 30 in DCS++ algorithm for record matching and the threshold value ranging from 0.5 to 0.9 with the gap of 0.1 is used in Levenshtein Distance algorithm for field matching of each record. If the user did not set these values, the system uses the default or best practice values such as 20 for window size and 0.7 for threshold value.

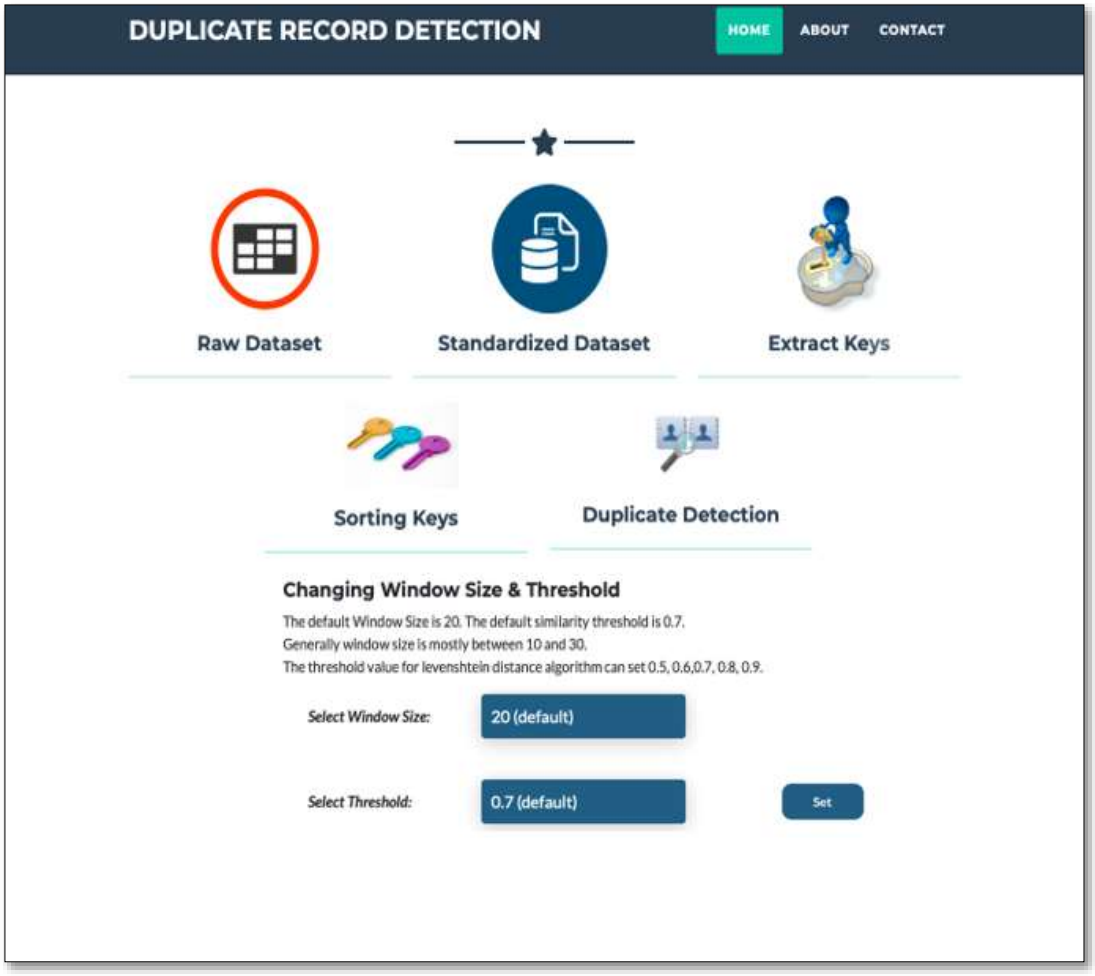


Figure 4.1 Main Section of the Proposed System

If the user clicks Raw Dataset Button from Home Page, the system shows the parsed records in a table form by parsing the data from XML Cora dataset into corresponding records. There are a total number of 1879 records in the dataset as shown in Figure 4.2.

Figure 4.2 Raw Dataset in Table Form

If the user clicks the “View XML Cora Dataset” link in the raw dataset page, the system appears with the detailed description of XML Cora dataset in the document tree as shown in Figure 4.3.

Figure 4.3 Cora Publication Xml Dataset

If the user clicks the Standardized Dataset Button from the Home Page, the system standardizes the fields Name, Date and Title. Author Name is standardized as First and Last by extracting the only first author and formatting it into the first character of first name, (.) dot and last name only. Date field is standardized by extracting only the year. Title field must not be the empty string. Figure 4.4 shows the standardized records in a table form by standardizing the data from parsed Cora dataset.

Figure 4.4 Standardized Dataset

If you click the Extract Keys Button from the Home Page, the system presents the list of records with created keys. In order to do duplicate detection, first the key creation process must be performed. In the key creation process, the system creates a key for each record by combining the first letter of first name, three consonants of last name, last two numbers of date and four consonants of title values after preprocessing stage as shown in Figure 4.5. The system needs a key for each relevant record in detecting near duplicates.

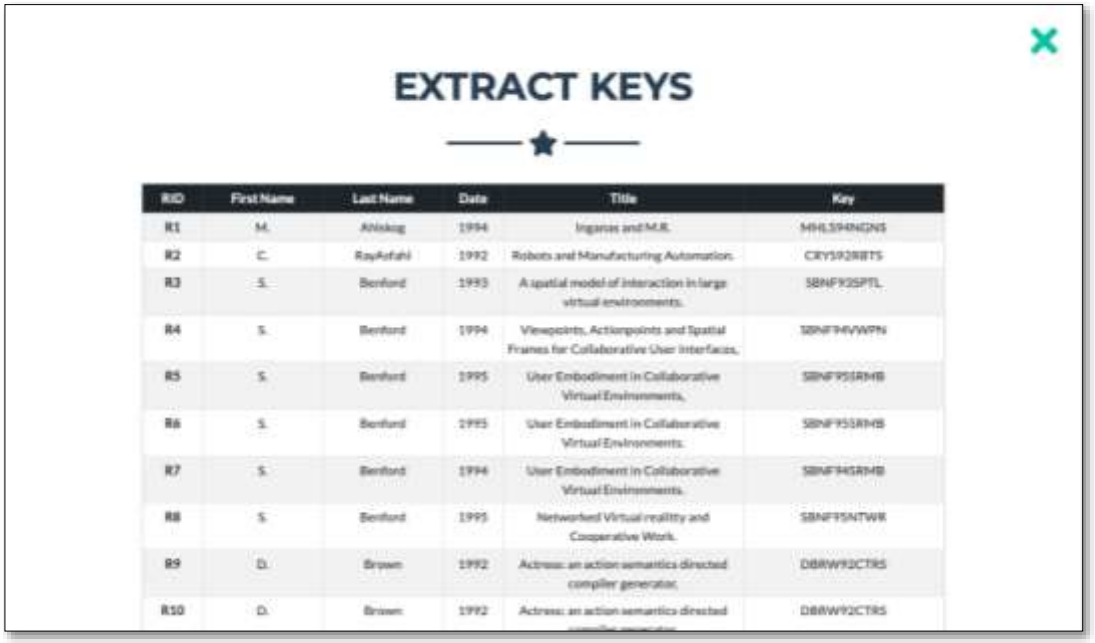


Figure 4.5 Key Creation

If the user clicks the Sorting Keys Button from the Home Page, the system shows the sorted record list by ordering the records according to the sorting keys which get from the key creation process. During the sorting phase, all matching or possible duplicate records perform close to each other in the record list as shown in Figure 4.6.

Figure 4.6 Sorting Keys

After sorting, a duplicate detection process can be performed. During the duplicate detection, the system uses the Levenshtein Distance algorithm to compute the similarity values between records. By using the DCS++ algorithm, this system reduces the number of record comparisons by skipping the windows for duplicates.

Five threshold values (0.5, 0.6, 0.7, 0.8, and 0.9) are used to test the duplicate records in cora publication XML dataset. The main purpose of this is to evaluate performance of this system. The similarity values of records are greater than threshold value, and then these records are duplicated and stored in a duplicate list. If the user wants to do duplicate detection, he/she can choose the desired threshold value and window size, and then the system will display the result according to the user selected window size and threshold as shown in Figure 4.7.

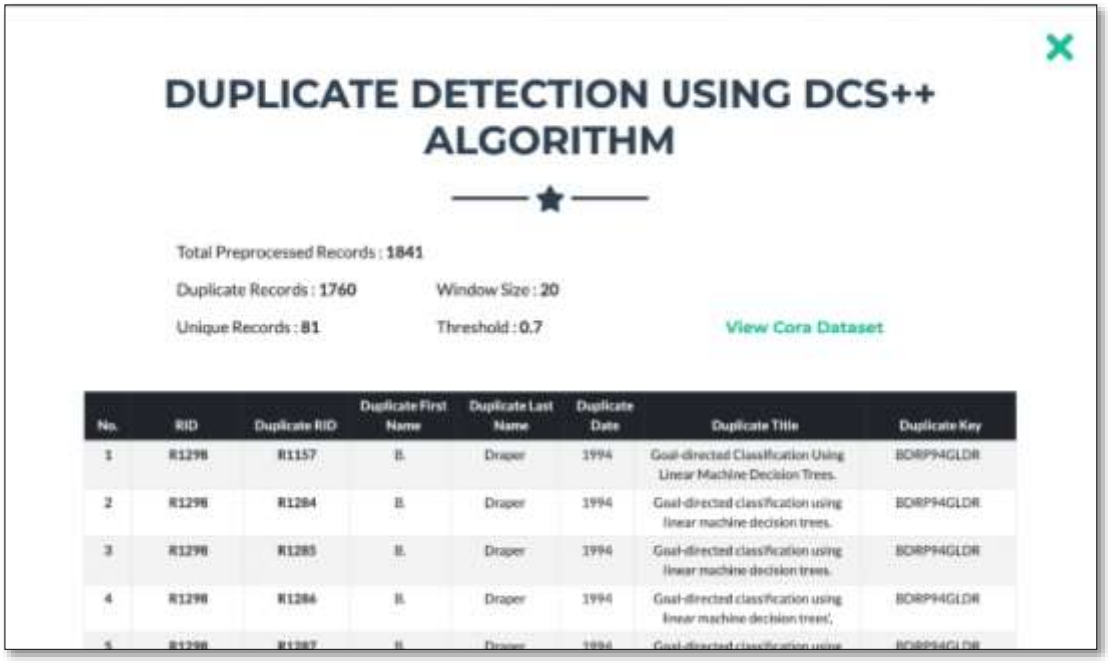


Figure 4.7 Duplicate Record Detection

4.3 Experimental Result

The proposed system evaluates the performance on Cora publication dataset which contains 1,879 records. The performance of the system is evaluated according to the percentages such as Recall, False positive error (FP), False negative error (FN) and Precision. The evaluation of the system is performed by ranging the threshold value from 0.5 to 0.9 with the gap of 0.1. The window size is set to be different window sizes 10, 20 and 30 for evaluation.

Figure 4.8, Figure 4.9 and Figure 4.10 show the execution results of the system for five thresholds (0.5, 0.6, 0.7, 0.8, 0.9) with different window sizes 10, 20 and 30.

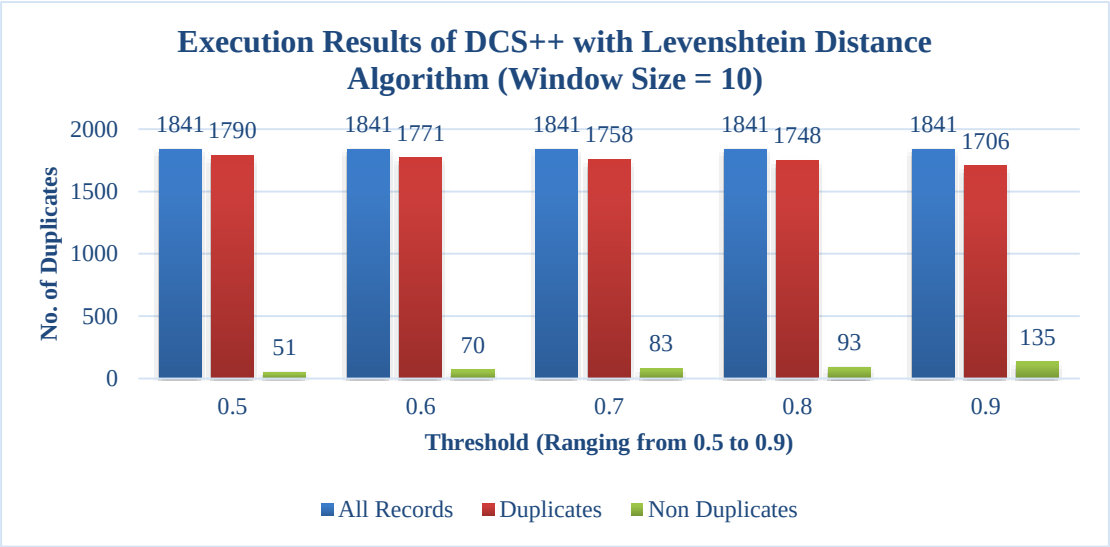


Figure 4.8 Execution Results with Window Size 10

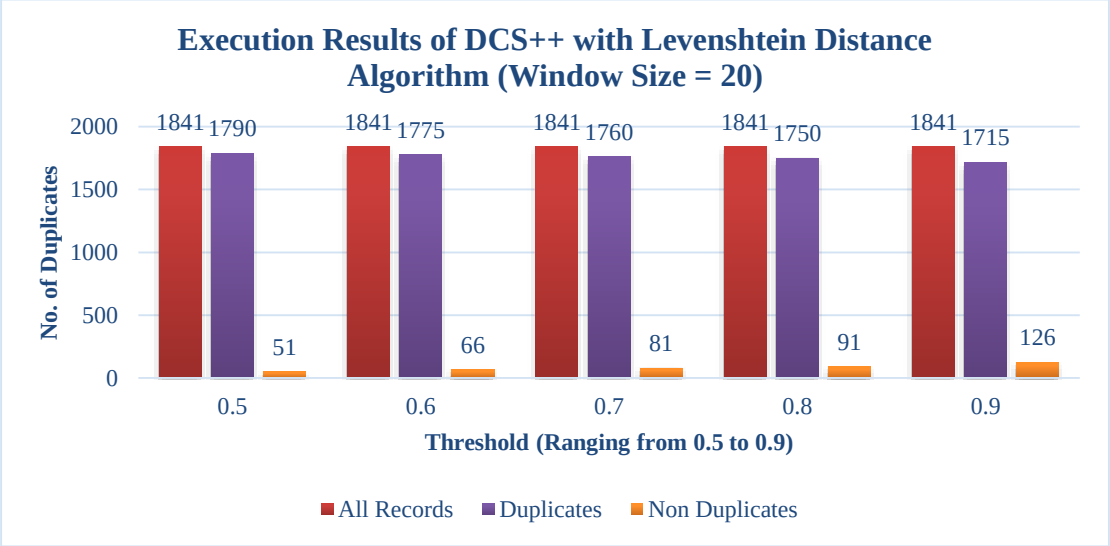


Figure 4.9 Execution Results with Window Size 20

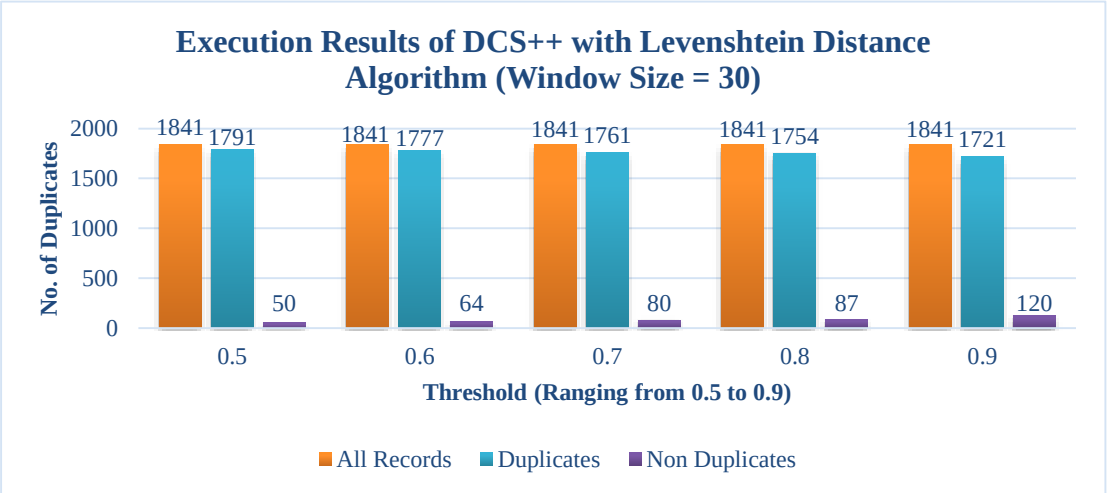
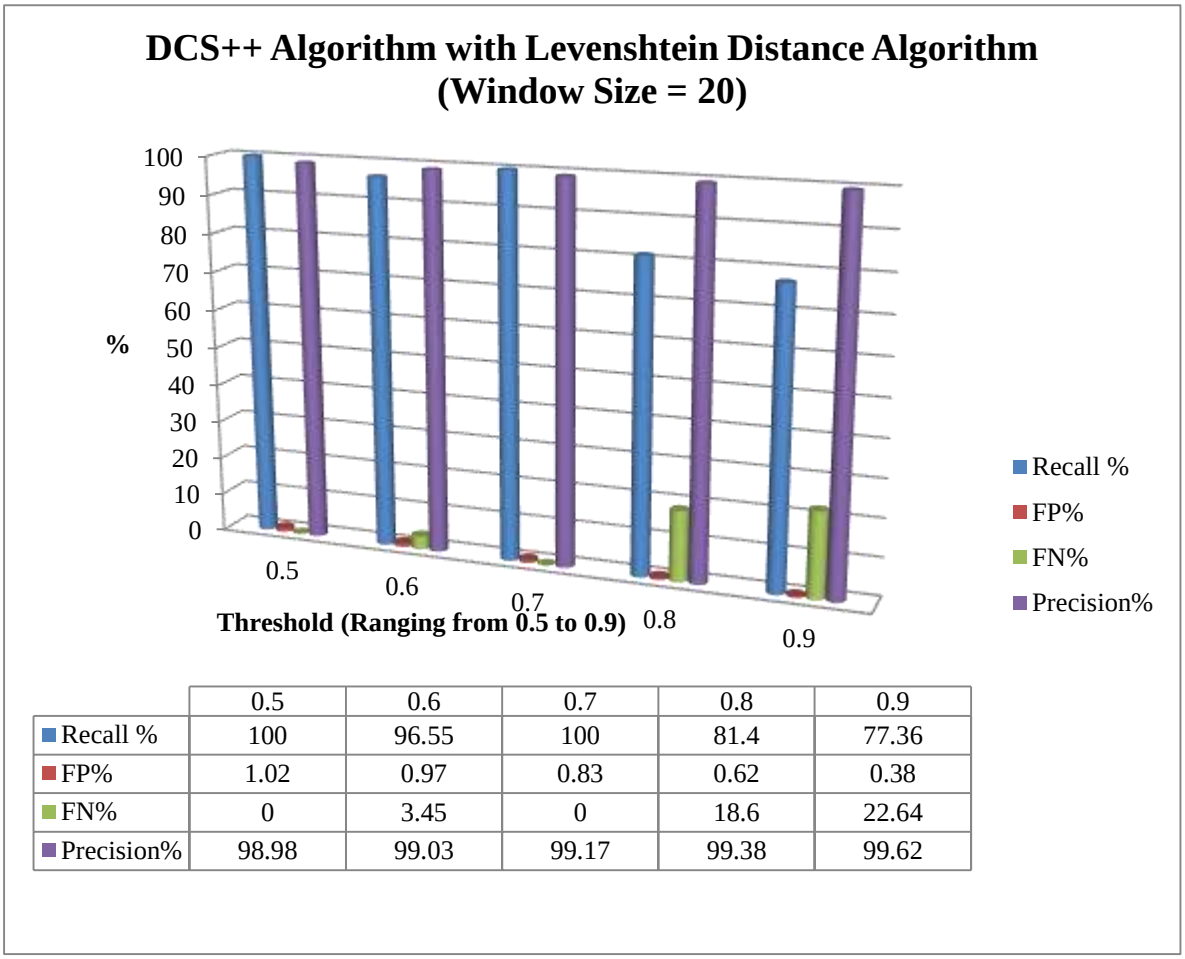


Figure 4.10 Execution Results with Window Size 30

Figure 4.11 shows the results of performance evaluation with the percentage of recall, FP, FN and precision by defining the initial window size 20 and five threshold values (0.5, 0.6, 0.7, 0.8, 0.9). In the proposed approach, the percentage values of recall and precision are high. Also, the percentage values of FP and FN are less in threshold values (0.5, 0.6, 0.7). Therefore, it determines that this system identifies duplicate records being correctly in threshold values (0.5, 0.6, 0.7). Although the percentage values of precision and FP are good in threshold values (0.8 and 0.9), other percentage values of recall and FN are not good because the percentage of duplicate records being correctly identified by the system is less.

The system assumes that the threshold values (0.5, 0.6 and 0.7) are better than other threshold values (0.8 and 0.9) for duplicate detection because these are less in the percentage of FP, FN and high in the percentage of precision and recall. Among them, we assume that threshold value 0.7 is the best result for duplicate detection of this system.



	0.5	0.6	0.7	0.8	0.9
Recall %	100	96.55	100	81.4	77.36
FP%	1.02	0.97	0.83	0.62	0.38
FN%	0	3.45	0	18.6	22.64
Precision%	98.98	99.03	99.17	99.38	99.62

Figure 4.11 Performance Evaluation of Duplicate Detection with Window Size 20

CHAPTER 5

CONCLUSION

In this system, DCS++ algorithm is used to detect possible duplicates in the cora publication of the dataset. Then, Levenshtein Distance algorithm is used for field matching of each record. The system was designed and implemented with PHP programming language on MacOS and Window platforms. The main objective of the system is to clean data for making business decisions and to reduce time and storage cost for data warehouses by using data cleaning methodology. Using the performance evaluation formulae, the evaluation of system performance is calculated based on the five threshold values. And the better result is the higher the percentage of precision and recall and then the less in the percentage of FP and FN. Therefore, using the results from the threshold values defined in the earlier table and chart, it can be concluded that the threshold value 0.7 is the best threshold value for duplicate detection in publication dataset using the DCS++ Algorithm. DCS ++ detects more duplicates by adding the next w-1 records of this duplicate to the window for each detected duplicate. This system has exceeded using a fixed window size. Time is critical in data cleaning of a large database. Usage of duplicate detection DCS++ method is to reduce the time taken on each comparison by skipping windows for duplicates. To sum up, by cleaning duplicate records in a dataset, it can be used effectively in decision making, query analysis and achieving high quality dataset. Benefits of the System are:

- improving the data quality,
- reducing the searching time for finding the desired data,
- reducing the extra space in memory due to record duplication and
- helping in mining the desired data easily.

5.1 Limitations and Further Extensions

There are some limitations in the proposed system. This system can only be used in a homogeneous source dataset such as XML format because it needs the key creation for some fields of the dependent domain and some changes are also required to reuse this algorithm for other datasets. If a user wants to test for other datasets rather than publication datasets, there is a separate process for parsing and normalization necessary. Therefore, it is limited to testing with a dataset from an independent domain

and it cannot handle the heterogeneous source of data. The goal of this system is to detect duplicate records in cora publication xml dataset. Other duplicate elimination systems can reference this system in the future as an improvement. Another area of future work lies in heterogeneous sources of publication datasets or domain-independent data cleaning.

AUTHOR'S PUBLICATION

[1] Yin Yin Phyo, Thidar Win, "Duplicate Record Detection in Data Cleaning Using DCS++ Algorithm", in the Proceedings of the (Paper ID: 80286, Accepted Date: 6th March 2020) Conference on Parallel and Soft Computing (PSC 2020), Yangon, Myanmar, 2020.

REFERENCES

[1] ZM Guo, AY Zhou, *"Research on data quality and data cleaning: A survey[J]"*, Journal of Software, 2002, 13(11): 2076-2082.

[2] Draisbach, Uwe and Felix Naumann, *"A Comparison and Generalization of Blocking and Windowing Algorithms for Duplicate Detection"*, 2009.

[3] 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, International Conference on, Wuhan, 2009.

[4] Q. Yang, Z. Guo, K. Wang, *"The SNM Algorithm Based on a Variety of Edit Distance and Variable Window"*, The 7th International Conference on Computer Engineering and Networks, 2017.

[5] Jumoke Soyemi, James Adegboye, *"Database Record Duplicate Detection System using Simil Algorithm,"* International Journal on Computer Science and Engineering (IJCSE),vol 10, 2018.

[6] P. A. V. Hall and G. R. Dowling, *"Approximate string matching"*, ACM Computing Surveys, 12(4):381-402, 1980.

[7] Abdulkhudhur, Hanan Najm, *"An improved Levenshtein algorithm for spelling correction word candidate list generation"*, 2016.

[8] S. B. Needleman and C. D. Wunsch, *"A general method applicable to the search for similarities in the amino acid sequence of two proteins"*, Journal of Molecular Biology. 48 (3): 443–53, 1970.

[9] Wang C, Yan RX, Wang XF, Si JN, Zhang Z, *"Comparison of linear gap penalties and profile-based variable gap penalties in profile-profile alignments"*, Comput Biol Chem. 35 (5): 308–318, October 2011.

[10] Smith, Temple F. and Waterman, Michael S., *"Identification of Common Molecular Subsequences"*, Journal of Molecular Biology. 147 (1): 195–197, 1981.

[11] Pinheiro, Jose C., and Don X. Sun, "*Methods for linking and mining massive heterogeneous databases."*, In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, pp. 309-313, 1998.

[12] Winkler, W. E.,*"Overview of Record Linkage and Current Research Directions"*, Research Report Series, RRS, 2006.

[13] Ukkonen, Esko, *“Approximate String Matching with q-grams and Maximal Matches.”*, Theoretical Computer Science, 1992.

[14] H. B. Newcombe, J. M. Kennedy. S. J. Axford and A.P. James, "Automatic linkage of vital records", Science, 130(3381):954-959, 1959.

[15] I P. Fellegi and A. B. Sunter, *"A theory for record linkage"*, Journal of the American Statistical Association, 64(328):1183-1210, 1969.

[16] Bachman, Philip & Sordoni, Alessandro & Trischler, Adam, *"Learning Algorithms for Active Learning"*, 2017.

[17] E. Monge and C. P. Elkan, *"The field matching problem: Algorithms and applications"*, In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), pages 267-270, 1996.

[18] Yu Jiang, Can Lin, Weiyi Meng, Clement Yu, Aaron M. Cohen, Neil R. Smalheiser, *"Rule-based deduplication of article records from bibliographic databases"*, Database, Volume 2014, 2014.

[19] Ravikumar, Pradeep; William Weston Cohen, *"A Hierarchical Graphical Model for Record Linkage"*, 20th Conference on Uncertainty in Artificial Intelligence, 2004.

[20] Bhattacharya, Indrajit; Lise Getoor, Latent Dirichlet, *“Allocation Model for Entity Resolution”*, Computer Science Department, University of Maryland, 2005.

[21] Hernández, Mauricio Antonio; Salvatore Joseph Stolfo, *"Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem"*, Data Mining and Knowledge Discovery 2 (1): 9--37, 1998.

[22] Kimball, Ralph; Joe Caserta, *"The data warehouse ETL toolkit: Practical techniques for extracting, cleaning, conforming, and delivering data"*, 2004.

[23] Hong T.P. ,Kuo C.S., Chi S. C., *"A fuzzy data mining algorithm for quantitative values"*, The Third International Conference on Knowledge Based Intelligent Information Engineering Systems.

[24] David Corrales, Agapito Ledezma, Juan Corrales. *“From Theory to Practice: A Data Quality Framework for Classification Tasks”*, Symmetry, 2018

[25] https://www.php.net

[26] https://hpi.de/naumann/projects/repeatability/datasets/cora-dataset.html

[27] https://www.apachefriends.org/index.html

APPENDIX A: OUTPUT DATA OF PROPOSED SYSTEM

In this section, the 20 objects of XML Cora dataset, the output of dataset in data preparation including data parsing, data standardization and the output of duplicates in duplicate record detection system are presented as the sample data of step by step procedures.

XML Cora Dataset

```
<CORA>
        <NEWREFERENCE id="1">
ahlskog1994a
                <author> M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M.
Fahlman, O. </author>
                <title> Inganas and M.R. </title>
                <journal> Andersson, J Appl. Phys., </journal>
                <volume> 76, </volume>
                <pages>893,</pages>
                <date> (1994). </date>
        </NEWREFERENCE>
        <NEWREFERENCE id="2">
ahlskog1994a
                <author> M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M.
Fahlman, O. Inganas and M.R.   Andersson, </author>
                <journal>  J Appl. Phys., </journal>
                <volume> 76, </volume>
                <pages>893, </pages>
                <date> (1994). </date>
        </NEWREFERENCE>
        <NEWREFERENCE id="3">
ahlskog1994a
                <author> M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M.
Fahlman, O.  Inganas and M.R.  Andersson, </author>
                <journal> J Appl. Phys.,</journal>
                <volume> 76, </volume>
                <pages>893, </pages>
                <date> (1994). </date>
        </NEWREFERENCE>
        <NEWREFERENCE id="4">
ahlskog1994a
                <author> M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M.
Fahlman, O. Inganas and M.R. Andersson,</author>
                <journal> J Appl. Phys., </journal>
                <volume> 76, </volume>
                <pages>893, </pages>
                <date> (1994). </date>
        </NEWREFERENCE>
        <NEWREFERENCE id="5">
```

```
ahlskog1994a
                <author> M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M.
Fahlman, O.  Inganas and M.R. Andersson,</author>
                <journal> J Appl. Phys., </journal>
                <volume> 76, </volume>
                <pages>893, </pages>
                <date> (1994). </date>
        </NEWREFERENCE>
        <NEWREFERENCE id="6">
ahlskog1994a
                <author> M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M.
Fahlman, O. Inganas and M.R. Andersson,</author>
                <journal> J Appl. Phys., </journal>
                <volume> 76, </volume>
                <pages>893, </pages>
                <date> (1994). </date>
        </NEWREFERENCE>
        <NEWREFERENCE id="7">
ahlskog1994a
                <author> M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M.
Fahlman, O. Inganas and M.R. Andersson,</author>
                <journal> J Appl. Phys., </journal>
                <volume> 76, </volume>
                <pages>893, </pages>
                <date> (1994). </date>
        </NEWREFERENCE>
        <NEWREFERENCE id="8">
ahlskog1994a
                <author> M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M.
Fahlman, O. Inganas and M.R. Andersson,</author>
                <journal> Journal of Applied Physics, </journal>
                <volume> 76, </volume>
                <pages>893, </pages>
                <date> (1994). </date>
        </NEWREFERENCE>
        <NEWREFERENCE id="9">
asfahl1992a
                <author> C. Ray Asfahl. </author>
                <title> Robots and Manufacturing Automation. </title>
                <publisher> John Wiley and Sons, </publisher>
                <address> New York, </address>
                <note> second edition, </note>
                <date> 1992. </date>
        </NEWREFERENCE>
        <NEWREFERENCE id="10">
benford1993a
                <author> Steve Benford and Lennart E. Fahlen. </author>
                <title> A spatial model of interaction in large virtual environments.
</title>
                <booktitle> In Proceedings of ECSCW'93, </booktitle>
```

```
                <address> Milan, </address>
                <date> 1993. </date>
        </NEWREFERENCE>
        <NEWREFERENCE id="11">
benford1994a
                <author> Benford, S., and Fahln, L. </author>
                <date> (1994), </date>
                <title> Viewpoints, Actionpoints and Spatial Frames for Collaborative
User Interfaces, </title>
                <booktitle> 6th ERCIM workshop, </booktitle>
                <date> June 1994, </date>
                <address>Stockholm. </address>
        </NEWREFERENCE>
        <NEWREFERENCE id="12">
benford1995a
                <author> Benford, S., Bowers, J., Fahln, L., Greenhalgh, C., and
Snowdon, D., </author>
                <title> User Embodiment in Collaborative Virtual Environments,
</title>
                <booktitle> in Proc. ACM Conference on Human Factors in
Computing Systems (CHI95), </booktitle>
                <date> May 7-11, 1995,</date>
                <address> Denver, Colorado, USA. </address>
        </NEWREFERENCE>
        <NEWREFERENCE id="13">
benford1995a
                <author> Benford, S., Bowers, J., Fahlen, L.E., Greenhalgh, C.,
Snowdon, D. </author>
                <date> (1995). </date>
                <title> User Embodiment in Collaborative Virtual Environments.
</title>
                <booktitle> In Proceedings of CHI95, </booktitle>
                <pages> 242-249. </pages>
        </NEWREFERENCE>
        <NEWREFERENCE id="14">
benford1995a
                <author> Benford, S., Bowers, J., Fahlen, L.E., Greenhalgh, C.,
Snowdon, D. </author>
                <title> User Embodiment in Collaborative Virtual Environments.
</title>
                <booktitle> In Proceedings of CHI95, </booktitle>
                <volume> 242 249. </volume>
                <date>1994.</date>
        </NEWREFERENCE>
        <NEWREFERENCE id="15">
benford1995b
                <author> Steve Benford, John Bowers, Lennart Fahlen, Chris
Greenhalg, John Mariani, and Tom Rodden. </author>
                <title> Networked Virtual realitty and Cooperative Work. </title>
                <journal> Presence,</journal>
```

```
                <volume> 4(4) </volume>
                <pages> 364-386, </pages>
                <date> 1995. </date>
        </NEWREFERENCE>
        <NEWREFERENCE id="16">
brown1992a
                <author> Brown, D. F., Moura, H. and Watt, D. A. </author>
                <date> (1992b), </date>
                <title> Actress: an action semantics directed compiler generator,
</title>
                <editor> in U. Kas-tens and P. Pfahler, eds, </editor>
                <booktitle> `Proceedings of the International Workshop on Compiler
Construction (CC-92)', </booktitle>
                <note>Vol. 641 of Lecture Notes in Computer Science, </note>
                <publisher> Springer-Verlag, </publisher>
                <address> Paderborn, Germany, </address>
                <pages> pp. 95-109. </pages>
        </NEWREFERENCE>
        <NEWREFERENCE id="17">
brown1992a
                <author> Brown, D. F., Moura, H. and Watt, D. A. </author>
                <date> (1992b), </date>
                <title> Actress: an action semantics directed compiler generator,
</title>
                <editor> in U. Kas-tens and P. Pfahler, eds, </editor>
                <booktitle> `Proceedings of the International Workshop on Compiler
Construction (CC-92)', </booktitle>
                <note>Vol. 641 of Lecture Notes in Computer Science, </note>
                <publisher> Springer-Verlag, </publisher>
                <address> Paderborn, Germany, </address>
                <pages> pp. 95-109. </pages>
        </NEWREFERENCE>
        <NEWREFERENCE id="18">
brown1992a
                <author> D. F. Brown, H. Moura, and D. A. Watt. Actress: </author>
                <title> an action semantics directed compiler generator. </title>
                <editor> In U. Kastens and P. Pfahler, editors, </editor>
                <booktitle> Proceedings of the 4th International Conference on
Compiler Construction (CC'92), </booktitle>
                <note>volume 641 of Lecture Notes in Computer Science, </note>
                <pages> pages 95-109, </pages>
                <address> Paderborn, FRG, </address>
                <date>October 1992. </date>
                <publisher> Springer-Verlag. </publisher>
        </NEWREFERENCE>
        <NEWREFERENCE id="19">
buth1992a 5.
                <author> B. Buth, K.-H. Buth, M. Franzle, B. v. Karger, Y.
Lakhneche, H. Langmaack, and M. Muller-Olm. </author>
```

```
                <title> Provably correct compiler development and implementation.
</title>
                <editor> In U. Kastens and P. Pfahler, editors, </editor>
                <booktitle> Compiler Construction, </booktitle>
                <note>volume 641 of Lecture Notes in Computer Science. </note>
                <publisher> Springer-Verlag, </publisher>
                <date> 1992. </date>
        </NEWREFERENCE>
        <NEWREFERENCE id="20">
buth1992a 6.
                <author> B. Buth et. al., </author>
                <date> 1992, </date>
                <title> Provably Correct Compiler Implementation, </title>
                <editor> in U. Karstens and P. Pfahler (eds.) </editor>
                <booktitle> Compiler Construction, </booktitle>
                <publisher> Springer Verlag, LNCS 641, </publisher>
                <pages> pp. 141-155. </pages>
        </NEWREFERENCE>
</CORA>
```

Parsed Dataset

RID	REF Name	Author	Title	Date	Booktitle	Address	Pages	Editor	Publisher	Tech	Institution	Journal	Volume	TitleDate	Note
R1	ahlskog1994a	M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O. Inganas and M.R.		(1994).			893,					Andersson, J Appl. Phys.,	76,		
R2	ahlskog1994a	M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O. Inganas and M.R. Andersson,		(1994).			893,					J Appl. Phys.,	76,		
R3	ahlskog1994a	M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O. Inganas and M.R. Andersson,		(1994).			893,					J Appl. Phys.,	76,		
R4	ahlskog1994a	M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M.		(1994).			893,					J Appl. Phys.,	76,		

		Fahlman, O. Inganas and M.R. Andersson,													
R5	ahlskog1994a	M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O. Inganas and M.R. Andersson,		(1994).			893,					J Appl. Phys.,	76,		
R6	ahlskog1994a	M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O. Inganas and M.R. Andersson,		(1994).			893,					J Appl. Phys.,	76,		
R7	ahlskog1994a	M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O. Inganas and M.R. Andersson,		(1994).			893,					J Appl. Phys.,	76,		

R8	ahlskog1994a	M. Ahlskog, J. Paloheimo, H. Stubb, P. Dyreklev, M. Fahlman, O. Inganas and M.R. Andersson,		(1994).			893,					Journal of Applied Physics,	76,			
R9	asfahl1992a	C. RayAsfahl.	Robots and Manufacturing Automation.	1992.		New York,			John Wiley and Sons,							second edition,
R10	benford1993a	Steve Benford and Lennart E. Fahlen.	A spatial model of interaction in large virtual environments.	1993.	In Proceedings of ECSCW'93,	Milan,										
R11	benford1994a	Benford, S., and Fahln, L.	Viewpoints, Actionpoints and Spatial Frames for Collaborative User Interfaces,	(1994),	6th ERCIM workshop,	Stockholm.										
R12	benford1995a	Benford, S., Bowers, J., Fahln, L., Greenhalgh, C., and Snowdon, D.,	User Embodiment in Collaborative Virtual Environments,	May 7-11, 1995,	in Proc. ACM Conference on Human Factors in Computing Systems (CHI95),	Denver, Colorado, USA.										

R13	benford1995a	Benford, S., Bowers, J., Fahlen, L.E., Greenhalgh, C., Snowdon, D.	User Embodiment in Collaborative Virtual Environments.	(1995).	In Proceedings of CHI95,		242-249.								
R14	benford1995a	Benford, S., Bowers, J., Fahlen, L.E., Greenhalgh, C., Snowdon, D.	User Embodiment in Collaborative Virtual Environments.	1994.	In Proceedings of CHI95,								242 249.		
R15	benford1995b	Steve Benford, John Bowers, Lennart Fahlen, Chris Greenhalg, John Mariani, and Tom Rodden.	Networked Virtual realitty and Cooperative Work.	1995.			364-386,					Presence,	4(4)		

R16	brown1992a	Brown, D. F., Moura, H. and Watt, D. A.	Actress: an action semantics directed compiler generator,	(1992b),	`Proceedings of the International Workshop on Compiler Construction (CC-92)',	Paderborn, Germany,	pp. 95-109.	in U. Kastens and P. Pfahler, eds,	Springer-Verlag,						Vol. 641 of Lecture Notes in Computer Science,
R17	brown1992a	Brown, D. F., Moura, H. and Watt, D. A.	Actress: an action semantics directed compiler generator,	(1992b),	`Proceedings of the International Workshop on Compiler Construction (CC-92)',	Paderborn, Germany,	pp. 95-109.	in U. Kastens and P. Pfahler, eds,	Springer-Verlag,						Vol. 641 of Lecture Notes in Computer Science,
R18	brown1992a	D. F. Brown, H. Moura, and D. A. Watt. Actress:	an action semantics directed compiler generator.	October 1992.	Proceedings of the 4th International Conference on Compiler Construction (CC'92),	Paderborn, FRG,	pages 95-109,	In U. Kastens and P. Pfahler, editors,	Springer-Verlag.						volume 641 of Lecture Notes in Computer Science,
R19	buth1992a5.	B. Buth, K.-H. Buth, M. Franzle, B. v. Karger, Y. Lakhneche, H. Langmaack, and M. Muller-Olm.	Provably correct compiler development and implementation.	1992.	Compiler Construction,			In U. Kastens and P. Pfahler, editors,	Springer-Verlag,						volume 641 of Lecture Notes in Computer Science.
R20	buth1992a6.	B. Buth et. al.,	Provably Correct Compiler Implementation,	1992,	Compiler Construction,		pp. 141-155.	in U. Karstens and P. Pfahler (eds.)	Springer Verlag, LNCS 641,						

Standardized Dataset

RID	First	Last	Date	Title	Booktitle	Address	Pages	Editor	Publisher	Tech	Institution	Journal	Volume	Titledate	Note
R1	M.	Ahlskog	1994	Inganas and M.R.			893,					Andersson, J Appl. Phys.,	76,		
R2	C.	RayAsfahl	1992	Robots and Manufacturing Automation.		New York,			John Wiley and Sons,						second edition,
R3	S.	Benford	1993	A spatial model of interaction in large virtual environments.	In Proceedings of ECSCW'93,	Milan,									
R4	S.	Benford	1994	Viewpoints, Actionpoints and Spatial Frames for Collaborative User Interfaces,	6th ERCIM workshop,	Stockholm.									
R5	S.	Benford	1995	User Embodiment in Collaborative Virtual Environments,	in Proc. ACM Conference on Human Factors in Computing Systems (CHI95),	Denver, Colorado, USA.									
R6	S.	Benford	1995	User Embodiment in Collaborative Virtual Environments.	In Proceedings of CHI95,		242-249.								
R7	S.	Benford	1994	User Embodiment in Collaborative Virtual Environments.	In Proceedings of CHI95,								242 249.		

R8	S.	Benford	1995	Networked Virtual realitty and Cooperative Work.			364-386,					Presence,	4(4)		
R9	D.	Brown	1992	Actress: an action semantics directed compiler generator,	`Proceedings of the International Workshop on Compiler Construction (CC-92)',	Paderborn, Germany,	pp. 95-109.	in U. Kas-tens and P. Pfahler, eds,	Springer-Verlag,						Vol. 641 of Lecture Notes in Computer Science,
R10	D.	Brown	1992	Actress: an action semantics directed compiler generator,	`Proceedings of the International Workshop on Compiler Construction (CC-92)',	Paderborn, Germany,	pp. 95-109.	in U. Kas-tens and P. Pfahler, eds,	Springer-Verlag,						Vol. 641 of Lecture Notes in Computer Science,
R11	D.	Brown	1992	an action semantics directed compiler generator.	Proceedings of the 4th International Conference on Compiler Construction (CC'92),	Paderborn, FRG,	pages 95-109,	In U. Kastens and P. Pfahler, editors,	Springer-Verlag.						volume 641 of Lecture Notes in Computer Science,
R12	B.	Buth	1992	Provably correct compiler development and implementation.	Compiler Construction,			In U. Kastens and P. Pfahler, editors,	Springer-Verlag,						volume 641 of Lecture Notes in Computer Science.
R13	B.	Buth et	1992	Provably Correct Compiler Implementation,	Compiler Construction,		pp. 141-155.	in U. Karstens and P. Pfahler (eds.)	Springer Verlag, LNCS 641,						

R14	B.	Buth et	1992	Provably Correct Compiler Implementation,	Compiler Construction,		pp. 141-155.	in U. Karstens and P. Pfahler (eds.)	Springer Verlag, LNCS 641,						
R15	B.	Buth et	1992	Provably Correct Compiler Implementation,	Compiler Construction,		pp. 141-155.	in U. Karstens and P. Pfahler (eds.)	Springer Verlag, LNCS 641,						
R16	B.	Buth	1992	Provably correct compiler development and implementation.	Compiler Construction, number 641 in Lecture Notes in Computer Science,		pages 141-155.	In U. Kastens and P. Pfahler, editors,	Springer-Verlag,						
R17	C.	Carlson	1993	Integrated CSCW Tools Within a Shared 3D Virtual Environment.	In Computer Supported Cooperative Work: A Book of Readings.	NY:	513	Ed. Irene Greit:	Morgan Kaufmann.						
R18	N.	Cramer	1985	A representation for the adaptive generation of simple sequential programs.	Proceedings of an International Conference on Genetic Algorithms and Their Applications,	Hillsdale, NJ.	pages 183-187,	In Grefenstette, J., editor,	Lawrence Erlbaum Associates.						

R19	W.	Daelemans	1995	IG-tree: A variant of IBL. submitted. Available from request to antal@cs.rulimburg.nl Fahlman,		Pittsburgh, PA.		S. E. and Lebiere, C.		Technical Report CMU-CS-90-100,	School of Computer Science, Carnegie-Mellon University,				
R20	W.	Daelemans	1989	An investigation of niche and species formation in genetic function optimization.	Proceedings of the Third International Conference on Genetic Algorithms,		pp. 42-50.	K. Deb and D. Goldberg	Morgan Kaufmann.						

Extract Keys

RID	First Name	Last Name	Date	Title	Key
R1	M.	Ahlskog	1994	Inganas and M.R.	MHLS94NGNS
R2	C.	RayAsfahl	1992	Robots and Manufacturing Automation.	CRYS92RBTS
R3	S.	Benford	1993	A spatial model of interaction in large virtual environments.	SBNF93SPTL
R4	S.	Benford	1994	Viewpoints, Actionpoints and Spatial Frames for Collaborative User Interfaces,	SBNF94VWPN
R5	S.	Benford	1995	User Embodiment in Collaborative Virtual Environments,	SBNF95SRMB
R6	S.	Benford	1995	User Embodiment in Collaborative Virtual Environments.	SBNF95SRMB
R7	S.	Benford	1994	User Embodiment in Collaborative Virtual Environments.	SBNF94SRMB
R8	S.	Benford	1995	Networked Virtual realitty and Cooperative Work.	SBNF95NTWR
R9	D.	Brown	1992	Actress: an action semantics directed compiler generator,	DBRW92CTRS
R10	D.	Brown	1992	Actress: an action semantics directed compiler generator,	DBRW92CTRS
R11	D.	Brown	1992	an action semantics directed compiler generator.	DBRW92NCTN
R12	B.	Buth	1992	Provably correct compiler development and implementation.	BBTH92PRVB
R13	B.	Buth et	1992	Provably Correct Compiler Implementation,	BBTH92PRVB
R14	B.	Buth et	1992	Provably Correct Compiler Implementation,	BBTH92PRVB
R15	B.	Buth et	1992	Provably Correct Compiler Implementation,	BBTH92PRVB
R16	B.	Buth	1992	Provably correct compiler development and implementation.	BBTH92PRVB
R17	C.	Carlson	1993	Integrated CSCW Tools Within a Shared 3D Virtual Environment.	CCRL93NTGR
R18	N.	Cramer	1985	A representation for the adaptive generation of simple sequential programs.	NCRM85RPRS
R19	W.	Daelemans	1995	IG-tree: A variant of IBL. submitted. Available from request to antal@cs.rulimburg.nl Fahlman,	WDLM95GTRV
R20	W.	Daelemans	1989	An investigation of niche and species formation in genetic function op-timization.	WDLM89NNVS

Sorting Keys

RID	First Name	Last Name	Date	Title	Sorting Key
R1298	A.	Bruce	1994	Goal-Directed Classification using Linear Machine Decision Trees.	ABRC94GLDR
R1246	A.	Cleeremans	1989	Finite state automata and simple recurrent networks.	ACLR89FNTS
R23	A.	Dempster	1977	Maximum likelihood from incomplete data via the EM algorithm.	ADMP77MXMM
R24	A.	Dempster	1977	Maximum likelihood from incomplete data via the EM algorithm.	ADMP77MXMM
R1282	A.	Danyluk	1993	Small Disjuncts in Action: Learning to Diagnose Errors in the Local Loop of the Telephone Network.	ADNY93SMLL
R1283	A.	Danyluk	1993	Small disjuncts in action: Learning to diagnose errors in the telephone network local loop,	ADNY93SMLL
R1000	A.	DW	1991	Instance-based Learning Methods,	ADW91NSTN
R1347	A.	Mehra	1997	"Self-parameterizing protocol stacks for guaranteed quality of service,"	AMHR97SLFP

R1346	A.	Mehra	1998	Self-parameterizing protocol stacks for guaranteed quality of service.	AMHR98SLFP
R1348	A.	Mehra	1998	"Self-parameterizing protocol stacks for guaranteed quality of service,"	AMHR98SLFP
R1349	A.	Mehra	1998	"Self-parameterizing protocol stacks for guaranteed quality of service,"	AMHR98SLFP
R1466	A.	Schwartz	1993	A reinforcement learning method for maximizing undiscounted rewards,	ASCH93RNFR
R1467	A.	Schwartz	1993	A reinforcement learning method for maximizing undiscounted rewards.	ASCH93RNFR
R12	B.	Buth	1992	Provably correct compiler development and implementation.	BBTH92PRVB
R13	B.	Buth et	1992	Provably Correct Compiler Implementation,	BBTH92PRVB
R14	B.	Buth et	1992	Provably Correct Compiler Implementation,	BBTH92PRVB
R15	B.	Buth et	1992	Provably Correct Compiler Implementation,	BBTH92PRVB

R16	B.	Buth	1992	Provably correct compiler development and implementation.	BBTH92PRVB
R1157	B.	Draper	1994	Goal-directed Classification Using Linear Machine Decision Trees.	BDRP94GLDR
R1284	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR

Duplicate Detection using DCS++ Algorithm

(Window Size = 20)

- Total Preprocessed Records: **1841**
- Duplicate Records: **1790**
- Unique Records: **51**
- Window Size: **20**
- Threshold: **0.5**

No.	RID	Duplicate RID	Duplicate First Name	Duplicate Last Name	Duplicate Date	Duplicate Title	Duplicate Key
1	**R1298**	**R1157**	B.	Draper	1994	Goal-directed Classification Using Linear Machine Decision Trees.	BDRP94GLDR
2	**R1298**	**R1284**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
3	**R1298**	**R1285**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
4	**R1298**	**R1286**	B.	Draper	1994	Goal-directed classification using linear machine decision trees',	BDRP94GLDR
5	**R1298**	**R1287**	B.	Draper	1994	Goal-directed classification using linear machine decision trees',	BDRP94GLDR
6	**R1298**	**R1288**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
7	**R1298**	**R1289**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
8	**R1298**	**R1290**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR

9	**R1298**	**R1291**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
10	**R1298**	**R1292**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR

- Total Preprocessed Records: **1841**
- Duplicate Records: **1775**
- Unique Records: **66**
- Window Size: **20**
- Threshold: **0.6**

No.	RID	Duplicate RID	Duplicate First Name	Duplicate Last Name	Duplicate Date	Duplicate Title	Duplicate Key
1	**R1298**	**R1157**	B.	Draper	1994	Goal-directed Classification Using Linear Machine Decision Trees.	BDRP94GLDR
2	**R1298**	**R1284**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
3	**R1298**	**R1285**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
4	**R1298**	**R1286**	B.	Draper	1994	Goal-directed classification using linear machine decision trees',	BDRP94GLDR
5	**R1298**	**R1287**	B.	Draper	1994	Goal-directed classification using linear machine decision trees',	BDRP94GLDR
6	**R1298**	**R1288**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
7	**R1298**	**R1289**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
8	**R1298**	**R1290**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR

9	**R1298**	**R1291**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
10	**R1298**	**R1292**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR

- Total Preprocessed Records: **1841**
- Duplicate Records: **1760**
- Unique Records: **81**
- Window Size: **20**
- Threshold: **0.7**

No.	RID	Duplicate RID	Duplicate First Name	Duplicate Last Name	Duplicate Date	Duplicate Title	Duplicate Key
1	**R1298**	**R1157**	B.	Draper	1994	Goal-directed Classification Using Linear Machine Decision Trees.	BDRP94GLDR
2	**R1298**	**R1284**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
3	**R1298**	**R1285**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
4	**R1298**	**R1286**	B.	Draper	1994	Goal-directed classification using linear machine decision trees',	BDRP94GLDR
5	**R1298**	**R1287**	B.	Draper	1994	Goal-directed classification using linear machine decision trees',	BDRP94GLDR
6	**R1298**	**R1288**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
7	**R1298**	**R1289**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
8	**R1298**	**R1290**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR

9	**R1298**	**R1291**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
10	**R1298**	**R1292**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR

- Total Preprocessed Records: **1841**
- Duplicate Records: **1750**
- Unique Records: **91**
- Window Size: **20**
- Threshold: **0.8**

No.	RID	Duplicate RID	Duplicate First Name	Duplicate Last Name	Duplicate Date	Duplicate Title	Duplicate Key
1	**R1298**	**R1157**	B.	Draper	1994	Goal-directed Classification Using Linear Machine Decision Trees.	BDRP94GLDR
2	**R1298**	**R1299**	B.	Draper	1994	Goal-directed Classification Using Linear Machine Decision Trees.	BDRP94GLDR
3	**R1298**	**R1300**	B.	Draper	1994	Goal-directed Classification Using Linear Machine Decision Trees.	BDRP94GLDR
4	**R1298**	**R1154**	C.	Brodley	1994	Goal-directed Classification Using Linear Machine Decision Trees.	CBRD94GLDR
5	**R1298**	**R1155**	C.	Brodley	1994	Goal-directed Classification Using Linear Machine Decision Trees.	CBRD94GLDR
6	**R1298**	**R1156**	C.	Brodley	1994	Goal-directed Classification Using Linear Machine Decision Trees.	CBRD94GLDR
7	**R1298**	**R1158**	C.	Brodley	1994	Goal-directed Classification Using Linear Machine Decision Trees.	CBRD94GLDR

8	**R23**	**R24**	A.	Dempster	1977	Maximum likelihood from incomplete data via the EM algorithm.	ADMP77MXMM
9	**R1282**	**R1283**	A.	Danyluk	1993	Small disjuncts in action: Learning to diagnose errors in the telephone network local loop,	ADNY93SMLL
10	**R1347**	**R1346**	A.	Mehra	1998	Self-parameterizing protocol stacks for guaranteed quality of service.	AMHR98SLFP

- Total Preprocessed Records: **1841**
- Duplicate Records: **1715**
- Unique Records: **126**
- Window Size: **20**
- Threshold: **0.9**

No.	RID	Duplicate RID	Duplicate First Name	Duplicate Last Name	Duplicate Date	Duplicate Title	Duplicate Key
1	**R23**	**R24**	A.	Dempster	1977	Maximum likelihood from incomplete data via the EM algorithm.	ADMP77MXMM
2	**R1347**	**R1346**	A.	Mehra	1998	Self-parameterizing protocol stacks for guaranteed quality of service.	AMHR98SLFP
3	**R1347**	**R1348**	A.	Mehra	1998	"Self-parameterizing protocol stacks for guaranteed quality of service,"	AMHR98SLFP
4	**R1347**	**R1349**	A.	Mehra	1998	"Self-parameterizing protocol stacks for guaranteed quality of service,"	AMHR98SLFP
5	**R1466**	**R1467**	A.	Schwartz	1993	A reinforcement learning method for maximizing undiscounted rewards.	ASCH93RNFR
6	**R12**	**R16**	B.	Buth	1992	Provably correct compiler development and implementation.	BBTH92PRVB
7	**R13**	**R14**	B.	Buth et	1992	Provably Correct Compiler Implementation,	BBTH92PRVB
8	**R13**	**R15**	B.	Buth et	1992	Provably Correct Compiler Implementation,	BBTH92PRVB

9	**R1157**	**R1284**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR
10	**R1157**	**R1285**	B.	Draper	1994	Goal-directed classification using linear machine decision trees.	BDRP94GLDR