International Journal of Engineering &Technology, 7 (4) (2018) 6175-6180

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v7i4.23127
Research paper



Duplicate detection and elimination in xml data for a data warehouse

Ghaith O. Mahdi [1] *, Murtadha M. Hamad [1]

[1] College of Computer Sciences and Information Technology, University of Anbar, Ramadi, Iraq
**Corresponding author E-mail: ghaithabdullah85@gmail.com*

Abstract

Due to the significant increase in the volume of data in recent decades, the problem of duplicate data has emerged because of the multiplicity of resources where the data is collected in different formats. The presence of duplicates comes as a result of the existence of different formulas of data. Thus, it is necessary to clean the duplicate data to access a pure data set. The main concern of this study is to clean data which Known by its complex hierarchal structure in data warehouse. This can be achieved by detecting duplicate in large data bases in order to increase the efficiency of data mining. In the current study the proposed system of duplicate elements passes through three-stages. The first stage (Pre-processing stage) includes two parts: the first part is the elimination of the exact match which, in turn, works to eliminate many of the identical elements completely. This procedure saves a lot of time and effort by preventing the entrance of many elements to the processing stage which are usually known by its complexity. In the second part blocking technique is used based on Levenshtein distance to minimize the number of comparisons and to maximize the accuracy of blocking elements than the traditional ones. These processes are performed to improve the dataset. The second stage (Processing stage) is taken to compute the similarity ratios between each pair of elements within each block by using smith waterman similarity algorithm. The third stage is the classification stage of the elements in which an element is identified whether it is duplicate or non-duplicate. The Artificial Neural Network technique (Back-Propagation) is used to meet this purpose. The threshold 0.65 has been determined which is relied on the results obtained. The Artificial Neural Network (Back-Propagation) is used to classify the elements in to duplicate and non-duplicate. The efficiency of the proposed system is represented by the accuracy obtained which is closer to 100% through reducing the number of "false negatives" and "false positive" relative to the "true positive".

Keywords: *Blocking Technique; Levenshtein Distance; Smith Waterman Similarity; ANN (Back Propagation)*

1. Introduction

XML is one of the popular kinds of data in data warehouses. Errors and inconsistences are usually present with such kind of data hence a need for XML data cleaning is so critical, this can be performed by identifying and eliminating duplicate [1]. Different representations of the same entity are given in a day thus they become a problem in the field of duplicate detection. A correct matching strategy is essentially used for identifying the duplicated entity because the same logical real world entity sometimes has multiple representations in the data warehouse. Identifying and removing duplicated data is not the sole problem in the comprehensive area of data cleaning and data quality in thedata warehouse [2] [3]. XML data hierarchically organized is semi-structured. The process of duplicate detection in XML data is labeled by its complication than well-structured data[4].
The process of detecting and eliminating duplicate records in hierarchical data is one of the important processes to support the concept of data cleaning and data integrating. The existence of more than one record of the same real world entity has a negative impact on the performance of operations in the data warehouse. Therefore, it is necessary to find a technique for detecting the duplicate records and delete it even if these records are not exactly identical.

2. Related works

In this part, we survey some previous studies on the detection of duplicates data and approaches to eliminate them. These approaches have close contact with our work.
S. Chaudhuri et. al. [5] improves an algorithm to eliminate duplicates in dimensional tables in data warehouse names DELPHI (Duplicate Elimination in the Presence of Hierarchies) that reduces the number of false positives without lost duplicates. Authors utilize dimensional hierarchy which contains a series of relations linked by key–foreign key is adopted to improve high quality duplicate elimination algorithm and then it evaluates on actual datasets from an operational data warehouses. final duplicate detection function is a weighted voting of the predictions from using co-occurrence similarity function and textual similarity function. DELPHI algorithm only applies to 1:N relationship. It cannot consider structure
F. Naumann et. al.[6] DogmatiX Tracks down Duplicates in XML. DogmatiX defines a general framework for identifying the duplicates. The records are checked whether they are duplicates or not based on their values. In real world, records are represented in several styles for the same object. The dogmatix framework is flexible to work on various algorithms and can add new methods to improve the framework. This framework consists of three types, Duplicate definition: Defines when two objects are duplicates.

Copyright ©2018 Ghaith O. Mahdi, Murtadha M. Hamad. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Candidate definition: Defines which document should be compare. Duplicate detection: Defines How Duplicates are searched. DogmatiX framework not good when dataset is too small or too large.
D. Milano et.al [7] suggested a method for measuring the distance of each XML data with one another, known as structure aware XML distance. Using the edit distance measure, similarity measure can be evaluated. This method compares only a portion of XML data tree whose structure is similar nature.
Akash R. Petkar and Vijay B. Patil [8]suggested an algorithm to define whether two records are duplicates or not depend on a given threshold. For finding the duplicates .The algorithm uses a Bayesian network. Called as Xml duplication using Xpath that requires little user interaction, since user just needs to provide the Xml dataset file and depend on that file the user has to provide the threshold value. The disadvantage in this technique is " High number of pairwise comparisons compromises efficiency".
Dr. Babasaheb Ambedkar [9]suggested the method that can find out similarities in the XML data elements. Bayesian network will determine the probability of two XML elements are similar. The Bayesian Network is used to configure elements structure which are being compared with each other and improved the efficiency and effectiveness. Also Check of typographical error in which two elements are compared by deleting its white spaces, also check for the other type of typographical error where spelling of any word is differing as per pronunciation of that individual. The Bayesian Network not focused on run time efficiency.
M. Bilenko et. al. [18] suggested adaptive framework for automatically learning blocking functions that are efficient and accurate. where describe two formulations based on learnable blocking functions and offer learning algorithms for training them. The effectiveness of the proposed methods is shown on real and simulated datasets, on which they verify to be more accurate than non-adaptive blocking methods. Disadvantage of this technique is the adaptation can increase or reduce number of comparisons, in other words, does not always ensure the reduction of the number of comparisons.

3. Literature review

3.1. Definition of duplication

Duplicate is defined as representing the same real entity in more than one form. The challenge of duplicate detection is to identify duplicate representation which are not precisely identical as a result of the existential mistakes in data such as typographical errors and spelling mistakes or there is no standard formula of data for example the European date and the American one are differently represented respectively like (day, month, year)(month/day/year). Thus, we require complex algorithms to calculate the similarity between pairs of records. Such algorithms are crucial for data cleansing and integration[10].

3.2. Blocking technique

The computation of similar records are specifically significant in linkage system. In addition to clustering, data mining task and schema mapping algorithms. This ascribes to the quadratic increasing growth of the object number with the size of dataset. It is impractical and prohibitive to calculate similarity between all pairs specially for large and complex dataset. Blocking methods are manipulated to alleviate this problem through computing the similarity within the same block solely. Consequently, useless comparisons are avoided. The blocking methods which are previously proposed manually construct an index through relying on similarity function and selecting set of predicates then they are followed by hand tuning of parameters. Efficiency is regarded as the great challenge in duplicate detection particularly when the dataset are huge. Comparing the record pairs which are not duplicated and similar unique entity keys available in all data bases will waste a lot of time and will be trivial. In most cases, no shared keys available between all records to be linked for example (A) and (B) thus each record from A must be compared with all records from B. The number of comparisons consequently come from A*B. For example A and B are 200,000 consecutively. Therefore, the total outcome will be (Forty billions) [11]. Blocking algorithms mean the use of some blocking keys to classify a set of records into disjoint blocks. Here the number of comparisons are highly minimized as a result of the comparisons within the same block. The good partitioning predicate should be carefully selected which determines the number and the size of partitions namely duplicate should be grouped in the same partition e.g. a typical partitioning of CRM application is zip code while if the same zip code available in same partition they are realized as duplicate records. Other partitioning may be grouped according to the last name or some fixed prefixes. The use of blocking technique effects execution time totally.
In some cases, the existence of erroneous values in certain attributes utilized to create the blocking keys lies behind grouping duplicate pairs in various blocks so it is no considered as duplicate[12].

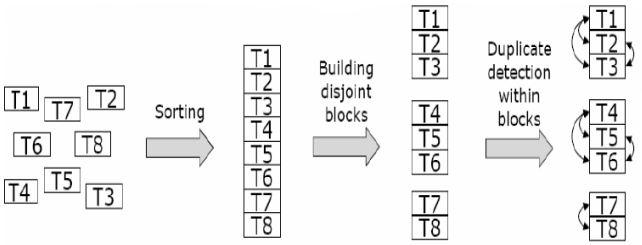


Fig. 1: Blocking Technique.

3.3. Levenshtein distance

Levenshtein distance is one of the prime algorithms which is utilized to measure the similarity between two strings: the first string is widely known as source string (s) and the second one is also known as a target string(t). The distance is the number of deletions, insertions, or substitutions which are lacked to transform s into t. The highly different strings are usually characterized by their having for greater Levenshtein distance[14].

Algorithm: Levenshtein Distance (Edit Distance)
Goal: Compute the distance between two strings
Input: Two string Source & Target
Output: Distance between Source & Target

1) Start
2) Input (Source, Target) Where Source = string1 & Target = string2
3) Compute the Source length & Target length
4) If (Source length=0)
5) Return (Target length) & Exit
6) Else if (Target length=0)
7) Return (Source length) & Exit
8) Else construct a matrix containing (0...Source length) as rows & (0. Target length) as columns
9) For i= 1: Source length
10) For j= 1: Target length
11) If Source[i] = Target[j],
12) put above diagonal value in the cell (d [i, j]:= d[i-1, j-1]).
13) If Source[i] <> Target[j]
14) Set cell d [i, j] of the matrix equal to the minimum of:

a) d [i-1, j] + 1.
b) d [i, j-1] + 1.
c) d [i-1, j-1] +1.

15) End For
16) End For
17) Return (Distance [source length, Target length])
18) End.

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

		S	a	t	u	r	d	a	y
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

Fig. 2: Show How the Levenshtein Distance Work in Simple Example.

3.4. Smith waterman algorithm

The Smith waterman similarity algorithm such the edit distance is defined the sequence of processes necessary to convert one string to another, but assigns lower weights to conversions between similar sounding characters and uses a specialized logic to deal with alignment gaps," i.e. there is a 'gap start' penalty corresponding to the beginning of a string of unmatched characters, and a separate 'gap continuation' penalty for its continuation". As with n-gram, similarities can be obtained through calculating the resulting value through the length of the shorter and the longer string, or the average length [15]. The Smith-Waterman similarity is a dynamic programming which gets the best local alignment between strings. The optimal local alignment acquired by the algorithm is accomplished in two phases. First, the alignment matrix is computed depend on the relationship between the characters. The optimal local alignment is initiated by finding the maximum element in the alignment matrix, which links the degree to the similarity between the two sequences, and tracing back the alignment matrix until a zero element is obtained [16]. When the local alignment is computed, a matrix $H_{i,j}$ is utilized to keep track of score of similarity between two sequences to be aligned (A_i and B_j). Each element of the matrix $H_{i,j}$ is computed according to the equation below:

$$H_{i,j} = \max \begin{cases} 0 \\ H_{i-1,j-1} + S_{i,j} \\ H_{i-1,j} - d \\ H_{i,j-1} - d \end{cases}$$

Where $S_{i,j}$ is the similarity score of comparing sequence A_i to sequence B_j and d is the penalty for a mismatch. The algorithm includes the three steps below:

a) Initialization step
b) Matrix fill step
c) Trace back step

The matrix is firstly initialized with $H_{i,0} = 0$ and $H_{0,j} = 0$, for all i and j. This is indicated as the initialization step. After the initialization, a matrix fill step is executed employing the above Equation, which includes all entries in the matrix. The trace back step is the final step, where the scores in the matrix are traced back to inspect for optimal local alignment. The trace back starts at the cell with the highest score in the matrix and continues up to the cell, where the score falls down to a predefined minimum threshold. In order to start the trace back, the algorithm requires to find the cell with the maximum value, which is done by traversing the entire matrix[17]. In this research we doesn't need to trace back phase. To compute the similarity score between two string we apply the following Equations:

$$Similarity\ score = \frac{max\ value\ in\ the\ matrix}{max\ distance\ between\ two\ string}$$

$$Max\ distance = min(lengthof\ string1, length\ of\ string2) * min(match\ value, gap\ value)$$

Algorithm: Smith waterman algorithm

Goal: Compute the similarity between two strings
Input: Two string str1, str2 and (match, mismatch, gap) value
Output: similarity score between str1 and str2

1) Start

Compute n, m where n= length (Str1), m=length (Str2)

2) For i= 1 to n
3) Return (n)
4) For j= 1 to m
5) Return (m)

Compute Max-distance

6) Max-distance=min (n, m) *max (match value, gap value)

Construct a matrix H and initialize the first column and first row. The matrix size is (n+1) *(m+1) .

7) For i= 1 to n
8) Matrix[i][0] = 0
9) For j= 1 to m
10) Matrix[0][j] =0
11) End Loop i
12) End Loop j

Filling the matrix.

13) For i= 1 to n
14) For j= 1 to m
15) letter1 = substr(Str1, j-1, 1)
16) letter2 = substr(Str2, i-1, 1)
17) If letter1 equal letter2
18) Diagonal value = Matrix[i-1] [j-1] + match value

Else

19) Diagonal value = Matrix[i-1] [j-1] +mismatch value
20) End If
21) Up value = Matrix[i-1] [j] + gap value
22) Left value = Matrix[i][j-1] + gap value
23) Current value= max (0, Diagonal value, Up value, Left value)
24) Matrix[i][j] = Current value
25) Current value=max (Current value, Matrix[i][j-1])
26) End Loop i
27) End Loop j

Compute the similarity score between str1 and str2

28) Similarity score= Current value/Max-distance
29) Return (similarity score)
30) End

Simple example: Compute the similarity score between GHA and GAS Use match value =+3, mismatch value= -3, gap value= 2

- ***Initialization phase***

		G	H	A
	0	0	0	0
G	0			
A	0			
S	0			

- *Filling the matrix*

		G	H	A
	0	0	0	0
G	0	3	1	0
A	0	1	0	4
S	0	0	0	2

- *Compute similarity score*

Max distance=min(3,3)*max(3,2)=9
Similarity score=4/9=0.5

4. Proposed method

In this section, we commence to manifest the basic phases of the proposed work "Duplicate records detection and eliminations". The parts of the system will be clarified in the following sub segment of this part with some illustrations and proposed calculations. In this paper, the blocking technique is developed to classify the XML data into groups. Traditional blocking technique cannot be used because XML data is semi-structured and there is no field in which the data is divide as it is in structured data

In this paper, we suggest utilize a blocking technique to improve the quality of grouping which is based on the text similarity calculation in the duplicate detection system. Furthermore, the quality of grouping is enhanced by averting any record which is not related to the block. This algorithm produces a most accurate group which has a closely related object by using Levenshtein distance. we split the dataset into several blocks depending on a linear equation to extract the first element as a base then we extract the rest of the base elements as well as Levenshtein distance algorithm is applied to calculate the distance between each element and all base elements. The distribution of the elements on the groups is relied on the minimum distance. After dividing the data set into groups, a similarity algorithm is used to calculate the similarity ratio between all elements within each group. This procedure does not only reduce the number of comparisons but also increases accuracy.

Smith waterman algorithm and artificial neural network (ANN) are used to detect and eliminate the duplicate record.

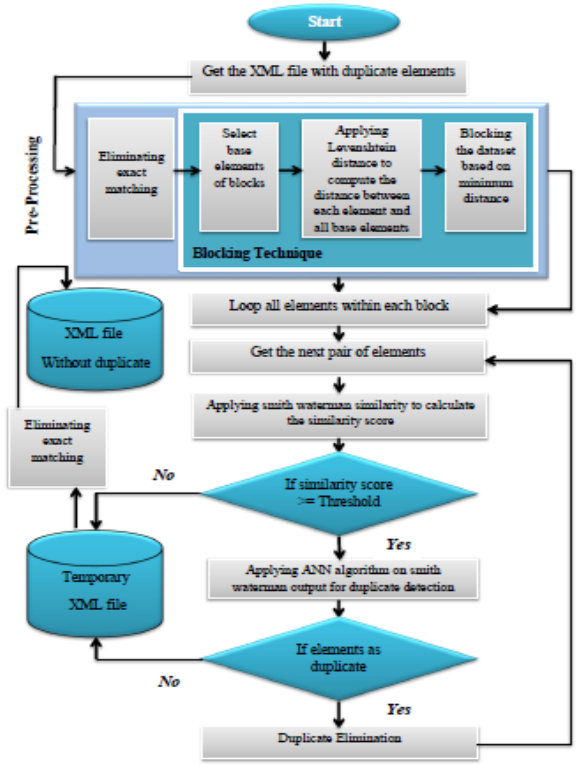


Fig. 4: Flowchart for Proposed Work.

5. Artificial neural network (ANN)

In this work, the researcher uses the Back-propagation (BP) Neural Network to perform the classification task of duplicate records by taking the similarity values for four attributes(publication-id, author, title, venue). The smith waterman algorithm is applied to compute the similarity values for the records within each block. The similarity values which are larger than the predefined threshold are sent to neural network to classify the Record as duplicate or not. Before this step, the similarity ratios of records are determined when the data is in comparison with other records within the same block. The Smith waterman similarity algorithm is employed to compare each of pair records in the same block. The output of this process contains percentages of matching. The researcher builds the artificial neural network that learns from the resulting values of the similarity algorithm previously mentioned. These values represent the similarity ratios of duplicated and non-duplicated elements where the artificial neural network is fed by a set of similarity values for the purpose of training. In this work, the neural network was fed by four similarity values and the target (+1) for positive states (duplicate element), similarity values and (-1) for negative states (non-duplicate element). The nutrition process continues until the neural network learns (stability stage) , In other words, when it reaches the stability stage, it can classify all the elements which are sent. The back-propagation algorithm was used to train the neural network in this work. For example: assuming the following two records fall within the same block:

First element of xml:

```
<publication id="fahlman1990a">
 <author id="117">S.E. Fahlman and C. Lebiere. </author>
 <title>The</title>
 <title>cascade-correlation</title>
 <title>learning</title>
 <title>architecture. </title>
 <venue>
 <venue pubid="fahlman1990a" id="29">
 <name>Advances in Neural Information Processing Systems</name>
            <vol>volume 2</vol>
            <date> 1990. </date>
 </venue>
 </venue>
 </publication>
```

Second element of xml:

```
<publication id=" fahlman1990">
 <author id="1">Aha</author>
 <author id="1"> D. W.</author>
 <author id="44"> Kibler</author>
 <author id="5"> D. and Albert</author>
 <title>"Instance-based</title>
 <title>learning</title>
 <title>algorithms",</title>
 <venue>
 <venue pubid=" fahlman1990" id="65">
            <name>Machine Learning 6(1)</name>
 <vol> 2</vol>
            <date> 1990. </date>
            </venue>
 </venue>
 </publication>
```

Table 1: Smith Waterman Similarity between Two Elements of Xml

	Publication- id	Authors	Title	Venue
(PM) of Xml1 and Xml2	95.28	15	17.64	70.04

Here, a neural network is initiated to learn similarity values from examples of duplicate and non-duplicate records. The above examples are fed to the neural network (Back Propagation). Thus, it can learn from them. Each example includes four values which represent similarity score and a target. The target is the proposed result (duplicate or not). As an example, if one regards the above two elements duplicate then the example consists of similarity values and the target (+1). Such example is called a positive example (duplicate elements). While the example which consists of similarity values and the target (-1) is called a negative example (non-duplicate elements). The feeding process continues until the neural network learns completely (stability stage). In other words, when it reaches the stability stage, it can classify all the elements which are sent.

Algorithm: ANN(Back-Propagation) algorithm.
Goal: Training the Neural Network back-propagation.
Input: Similarity score for four attribute
Output: Duplicate (1), non-duplicate(-1).

1) Begin
2) Create the network [input: 4 neurons, output: 1 neuron and activation function (log sigmoid)].
3) Set biases and weights as random value.
4) Apply the Back-propagation training.
5) Initialization epoch =1.
6) Train the data (input pair, target, and output).
7) Training the network.
8) Simulate the network.
9) If the output predicted equal actual output
   Display the result
   Else update biases and weight
   Go to step (6), epoch=epoch +1.
10) End begin

The Back Propagation neural network algorithm consists of multiple layers: the first layer uses for input units 4 neurons, the second layer (hidden layer) consists of 10 neurons, while the last layer (output layer) produces 1 neuron that represents the duplication state (duplicated and non-duplicated).
A result of smith waterman algorithm is exported to the Neural Network in order to classify the duplicated and non-duplicated elements depending on the training of the artificial neural network. The artificial neural network is trained by sampling comparison results of the dataset elements. In this method, the technique composes of two steps (training data and testing data). Figure 5 shows the best training performance at epoch 7, which characterizes the performance at the 5.3686e-09 and the gradient at 2.79e-06.

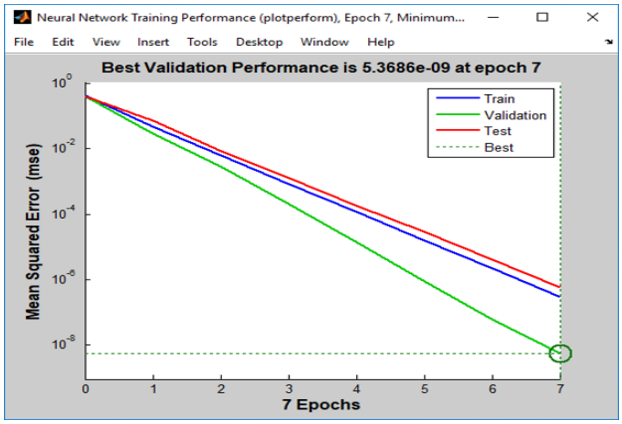


Fig. 5: Represent the Best Training Regression.

6. Experimental result

This section gives a concise depiction of the test performed utilizing the dataset. The test was done on the same set of data that is usually used in the process of detecting duplicates (Cora dataset).To measure the effectiveness of the proposed work three parameters are used: recall, precision and F-score. By execution the designed application and implementing the proposed algorithms. We obtained the outcomes which will be viewed in next table.
Recall is the ratio of the number of relevant records retrieved to the total number of relevant records in the database. It is usually expressed as a percentage.

$$\text{Recall(R)} = \text{TP}/(\text{TP} + \text{FP})$$

Precision is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. It is usually expressed as a percentage.

$$\text{Precision(P)} = \text{TP}/(\text{TP} + \text{FN})$$

F-Score is the harmonic mean of the precision and recall values.

$$\mathbf{F-Score} = \mathbf{2} * (\mathbf{P} * \mathbf{R})/(\mathbf{P} + \mathbf{R})$$

TP (True Positive) is the correct detection of duplicate FP (False Positive) is the record is not duplicate but detected as duplicate FN(False Negative) is the records that are duplicate but not detected as duplicate. The proposed work was implemented in Microsoft visual studio 2017 and MATLAB(R2012a). It was run on a PC with an Intel(R) core (TM) processor and 8GB RAM and the operating system is windows 10 professional.

Table 2: Performance Achieved Using Proposed Method on Real Dataset

File	Precision	Recall	F-Score
Cora dataset	98.5%	97.8%	98.1%

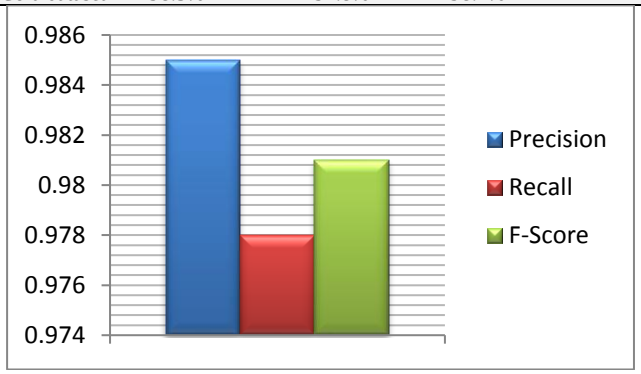


Chart 1: Performance of Proposed Work.

To measure the effectiveness of the proposed method it is compared with XMLDup and Dogmatix.

Table 3: Comparison of Results of Dogmatix, Xmldup and Proposed Work

File	Proposed work		Dogmatix		XMLDup	
Cora Dataset	P	R	P	R	P	R
	0.98	0.97	0.81	0.93	0.87	0.99

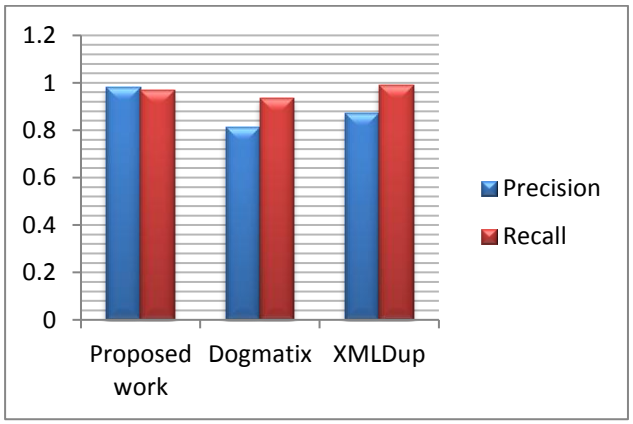


Chart 2: Comparison of Results.

7. Conclusion and future work

In this paper, we introduced a new effective method of grouping data that improved the quality of grouping based on the Levenshtein distance. The new method offers high accuracy in blocking the data closest to each other and this leads to avoiding unnecessary comparison. We also used artificial neural network technique (Back-Propagation) technique in the classification of duplicated elements and this method achieved high efficiency in terms of precision and recall. In future work, we want to use cheap similarity function such as cosine similarity for blocking the dataset to save time and another technique to classify duplicate elements in XML data such as the decision tree.

References

[1] M. R. Pawar, "Efficient Duplicate Detection and Elimination in Hierarchical Multimedia Data," vol. 122, no. 12, pp. 15–21, 2015. https://doi.org/10.5120/21751-5018.

[2] A. A. Abraham and S. D. Kanmani, "A Novel Approach for the Effective Detection of Duplicates in XML Data," *Int. J. Comput. Eng. Res.*, vol. 4, pp. 82–87, 2014.

[3] M. M. Hamad and S. S. Sami, "Using Q-Gram and Fuzzy Logic Algorithms for Eliminating Data Warehouse Duplications," 2016.

[4] S. Gaikwad and N. Bogiri, "Levenshtein distance algorithm for efficient and effective XML duplicate detection," *IEEE Int. Conf. Comput. Commun. Control. IC4 2015*, 2016. https://doi.org/10.1109/IC4.2015.7375698.

[5] R. Ananthakrishna, S. Chaudhuri, and V. Ganti, "Eliminating Fuzzy Duplicates in Data Warehouses," *VLDB '02 Proc. 28th Int. Conf. Very Large Databases*, pp. 586–597, 2002. https://doi.org/10.1016/B978-155860869-6/50058-5.

[6] M. Weis and F. Naumann, "DogmatiX Tracks down Duplicates in XML," *Proc. 2005 ACM SIGMOD Int. Conf. Manag. data,ACM*, pp. 431–442, 2005. https://doi.org/10.1145/1066157.1066207.

[7] L. Leitão, P. Calado, and M. Weis, "Structure-based inference of xml similarity for fuzzy duplicate detection," *16th ACM Conf. Inf. Knowl. Manag.*, pp. 293–302, 2007. https://doi.org/10.1145/1321440.1321483.

[8] A. R. Petkar and V. B. Patil, "Duplicate Detection in Hierarchical Data Using XPath," *IOSR J. Comput. Eng. Ver. I*, vol. 17, no. 6, pp. 2278–661, 2015.

[9] A. N. Mehta, "Similarity Detection for XML Data," *Int. J. Adv. Reserch Sci. Eng.*, vol. 5, no. 1, pp. 152–157, 2016.

[10] P. B. K. P. M. Bhavana Dhake1, Dr.S.S.Lomte2, Prof.Y.R.Nagargoje3,Prof.R.A.Auti4,"DuplicatDetection in Hierarchical Data Using Improved Network Pruning Algorithm," *Compusoft*, vol.4, no. 6, pp. 7838–7850, 2015.

[11] A. Thesis, "Performance Evaluation of Blocking Methods for Duplicate Record Detection," 2010.

[12] U. Draisbach and F. Naumann, "A generalization of blocking and windowing algorithms for duplicate detection," *Proc. - 2011 Int. Conf. Data Knowl. Eng. ICDKE 2011*, pp. 18–24, 2011. https://doi.org/10.1109/ICDKE.2011.6053920.

[13] J.ARUNA, "Identification of Duplication Records For Query Results from Real Time Databases," B.S.Abdur Rahman University, 2012.

[14] R. Haldar and D. Mukhopadhyay, "Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach," *arXiv:1101.1232*, no. Ld, pp. 286–293, 2011.

[15] G. Recchia and M. Louwerse, "A Comparison of String Similarity Measures for Toponym Matching," no. c, 2013.

[16] C. Ling, K. Benkrid, and T. Hamada, "A parameterisable and scalable smith-Waterman algorithm implementation on CUDA-compatible GPUs," *2009 IEEE 7th Symp. Appl. Specif. Process. SASP 2009*, pp. 94–100, 2009. https://doi.org/10.1109/SASP.2009.5226343.

[17] L. Hasan, Z. Al-Ars, and Z. Nawaz, "A Novel Approach for Accelerating the Smith-Waterman Algorithm using Recursive Variable Expansion," *Proc. 19th Annu. ...*, 2008. https://doi.org/10.1109/IDT.2008.4802483.

[18] M. Bilenko, M. View, and R. J. Mooney, "Adaptive Blocking : Learning to Scale Up Record Linkage," Proc. Sixth IEEE Int. Conf. Data Min., no. December, pp. 87–96, 2006. https://doi.org/10.1109/ICDM.2006.13.