Duplicate Records Elimination in Bibliographical Dataset using Priority Queue Algorithm with Smith-Waterman Algorithm

Su Mon Thaung, Thin Thin Htike
University of Computer Studies, Pathein
sumonthaung99@gmail.com

Abstract

Often, in the real world, entities have two or more representations in databases. Duplicate records do not share a common key and / or they contain errors that make duplicate matching a difficult task. A major problem that arises from integrating different databases is the existence of duplicates. Data cleaning is the process for identifying two or more records within the database, which represent the same real world object (duplicates), so that a unique representation for each object is adopted. This system addresses the data cleaning problem of detecting duplicate records that are approximate duplicates, but not exact duplicates. It uses Priority Queue algorithm with Smith Waterman algorithm for computing minimum edit-distance similarity values to recognize pairs of approximately duplicates and then eliminate the detected duplicate records. And, we also determine the performance evaluation with the lowest FP %(false positive percentage) and FN %(false negative percentage) as the best result.

1. Introduction

Databases play an important role in today's IT based economy. Many industries and systems depend on the accuracy of databases to carry out operations. To improve the data quality [7], data cleaning is especially required when integrating heterogeneous data sources. One of the most intriguing data quality problems is that of multiple, yet different representations of the same real world object in the data.

The process of detecting and removing database defects and duplicates is referred to as data cleaning. Duplicate elimination is hard because it is caused by different types of errors like typographical errors, missing values, abbreviations and different representations of the same logical value. In order to provide access to accurate and consistent data, consolidation of different data representations and elimination of duplicate information become necessary.

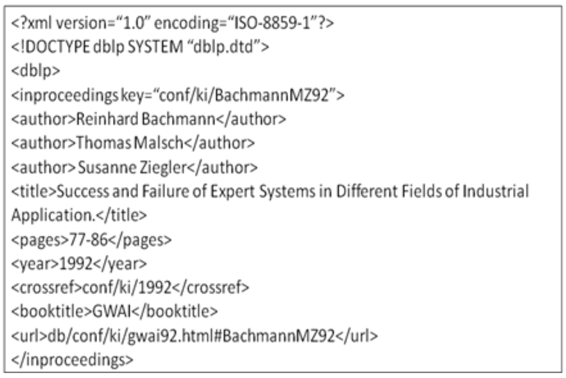
2. Related Works

The duplicate detection problem is different from, but related to, the schema matching problem [2, 6]. The problem of actually detecting matching records still exists even when the schema matching problem has been solved.

Record matching may also be used as a substitute for detailed schema matching, which may be impossible for semi-structured data. In general, we are interested in situations where several records may refer to the same real-world entity, while is not being syntactically equivalent. A set of records that refer to the same entity can be interpreted in two ways. One way is to view one of the records as correct and the other records as duplicates containing erroneous information. The task then is to cleanse the database of the duplicate records [4, 5]. Another interpretation is to consider each matching record as a partial source of information.

3. Overview of the System

In this system, we use the bibliographical XML dataset from the DBLP repository which contains 860 records. DBLP (Digital Bibliography & Library Project) is an internet "newcomer" that started service in 1993. Figure 1 shows the example of bibliographical XML dataset.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE dblp SYSTEM "dblp.dtd">
<dblp>
<inproceedings key="conf/ki/BachmannMZ92">
<author>Reinhard Bachmann</author>
<author>Thomas Malsch</author>
<author>Susanne Ziegler</author>
<title>Success and Failure of Expert Systems in Different Fields of Industrial
Application.</title>
<pages>77-86</pages>
<year>1992</year>
<crossref>conf/ki/1992</crossref>
<booktitle>GWAI</booktitle>
<url>db/conf/ki/gwai92.html#BachmannMZ92</url>
</inproceedings>
```

Figure 1. Example of Bibliographical XML Dataset

The DBLP service evolved from a small bibliography specialized to database systems and logic programming to a digital library covering most subfields of computer science. This service is available from several hosts (University of Trier), CS departments (Trier, Germany), ACM SIGMOD (New York, USA), VLDB Endowment, Sunsite Central Europe (Aachen, Germany).

The major contribution of the system examines the extent for redundancy in bibliographical XML data from DBLP database using priority queue algorithm with smith-waterman algorithm. The overview of the system is shown in Figure 2.

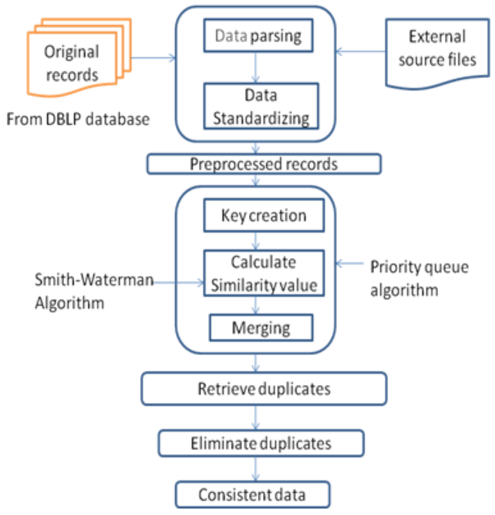


Figure 2. Overview of the System

It uses bibliographical XML dataset in large scale DBLP database as input. We are commonly divided into three stages in our proposed system.

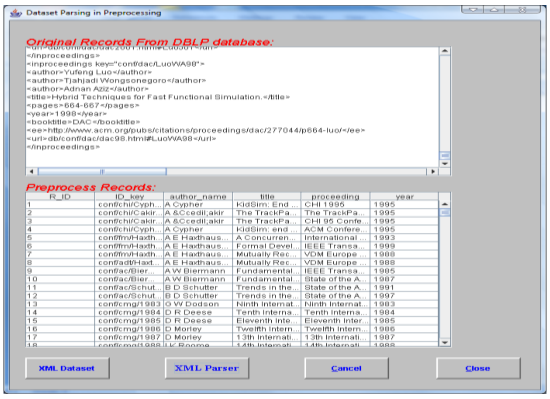
In preprocessing stage, we parsing the bibliographical record set in XML format from DBLP repository into each fields. We use priority queue algorithm and smith-waterman algorithm in duplicate detection stage. In duplicate elimination stage, only one record is retained and eliminated other duplicates.

3.1. Preprocessing

Data cleaning is an important preprocessing step before duplicate detection. We perform parsing and standardizing stage for preprocessing.

3.1.1. Data Parsing

Parsing locates, identifies and isolates individual data elements in the source files. It uses XML bibliographic dataset as input and parsing them to detect syntax errors. Parsing XML dataset is shown in Figure 3.



Original Records From DBLP database:

```
</inproceedings>
<inproceedings key="conf/dac/LuoWA98">
<author>Yufeng Luo</author>
<author>Tjahjadi Wongsonegoro</author>
<author>Adnan Aziz</author>
<title>Hybrid Techniques for Fast Functional Simulation.</title>
<pages>664-667</pages>
<year>1998</year>
<booktitle>DAC</booktitle>
<ee>http://www.acm.org/pubs/citations/proceedings/dac/277044/p664-luo/</ee>
<url>db/conf/dac/dac98.html#LuoWA98</url>
</inproceedings>
```

Preprocess Records:

R_ID	ID_key	author_name	title	proceeding	year
1	conf/chi/Cyph...	A Cypher	KidSim: End ...	CHI 1995	1995
2	conf/chi/Cakir...	A &Ccedil;akir	The TrackPa...	The TrackPa...	1995
3	conf/chi/Cakir...	A &Ccedil;akir	The TrackPa...	CHI 95 Confe...	1995
4	conf/chi/Cyph...	A Cypher	KidSim: end ...	ACM Confere...	1995
5	conf/fm/Haxth...	A E Haxthaus...	A Concurren...	International ...	1993
6	conf/fm/Haxth...	A E Haxthaus...	Formal Devel...	IEEE Transa...	1999
7	conf/fm/Haxth...	A E Haxthaus...	Mutually Rec...	VDM Europe ...	1988
8	conf/adt/Haxt...	A E Haxthaus...	Mutually Rec...	VDM Europe ...	1988
9	conf/ac/Bier...	A W Biermann	Fundamental...	IEEE Transa...	1985
10	conf/ac/Bier...	A W Biermann	Fundamental...	State of the A...	1987
11	conf/ac/Schut...	B D Schutter	Trends in the...	State of the A...	1991
12	conf/ac/Schut...	B D Schutter	Trends in the...	State of the A...	1997
13	conf/cmg/1983	G W Dodson	Ninth Internat...	Ninth Internat...	1983
14	conf/cmg/1984	D R Deese	Tenth Interna...	Tenth Interna...	1984
15	conf/cmg/1985	D R Deese	Eleventh Inte...	Eleventh Inter...	1985
16	conf/cmg/1986	D Morley	Twelfth Intern...	Twelfth Intern...	1986
17	conf/cmg/1987	D Morley	13th Internati...	13th Internati...	1987

Figure 3. Parsed Bibliographical XML Dataset

3.1.2. Data Standardizing

Data standardization refers to the process of standardizing the information represented in certain fields to a specific content format. It makes the author-name fields in parsed bibliographical dataset to a specific content format. Table 1 shows example of standardize dirty data field in preprocessing.

Table 1. Example of Standardized Dirty Data Field

Author Names	**Standardized Author Names**
Robert S. Arnold	R. S. Arnold
H. Raymond Strong	H. R. Strong
Frank Manola	F. Manola
Pieter Unema	P. Unema
Edward A. Macnair	E. A. Macnair

3.2. Duplicate Detection

In duplicate detection stage, these preprocessed bibliographical records become the input to this stage. Firstly, key creation process must be performed to sort the dataset and to help duplicate detection easily. After that, these sorted records are the inputs to priority queue algorithm for merging. We intend to use priority queue algorithm for faster identification. These record pairs are then computed their similarity values using smith-waterman algorithm. The similarity values are compared with certain threshold value. If the comparison succeeds, we can get duplicate record pairs.

3.2.1. Key Creation

During key creation process, a key is computed for each record in the database by extracting relevant fields or portions of fields which form an important discriminating attribute [9]. Table 2 shows example of performing key creation process.

Table 2. Example of Key Creation Process

First author		Second author		Year	Key
First	Last	First	Last		
H	Petrie	S	Morley	1973	PTRHM RLS73
J	Gray	V	Watson	1975	GRYJW TSV75
M	Fried	J	Rosen	1978	FRDMR SNJ78

3.2.2. Sorting

The records in the database are sorted using the keys computed in key creation process.

3.2.3. Merging

After the parsed Bibliographical records have been sorted, we use priority queue through the sequential list of records limiting the comparisons for record matching. We intend to use priority queue algorithm to improve the efficiency of duplicate records detection.

3.2.3.1. Overall Priority Queue Algorithm

The algorithm scans the sorted database with a priority queue of record subsets belonging to the last few clusters detected. The priority queue contains a fixed number of sets of record. In this experiment, this number is 10.

It scans through the sorted dataset sequentially. Suppose that record R_j is the record currently being considered. It first tests whether R_j is already known to be a member in priority queue. If R_j contains in priority queue, we continue with the next record, R_{j+1}.

On the other hand, R_j is not contain in priority queue, it uses the smith-waterman algorithm to compare R_j with records in the priority queue. It iterates through each record in priority queue, starting with the highest priority record. If similarity values are greater than a certain threshold, then store these record pairs in duplicate table. If no match found, R_j is compared with the next record in priority queue.

Finally, if R_j is compared to records in priority queue without detecting that it is a duplicate of any of these, R_j is saved in the priority queue with highest priority. If this action causes the size of the priority queue to exceed its limit then the lowest priority record is removed from the priority queue. Then, we scan the next record R_{j+1} in database until scanning all records in database and process the above steps again and again.

Instead of comparing every record with every other record, we place a set of records that is suspected to contain duplicates in priority queue. Then it compares every record with records in priority queue. This doing reduces the number of record comparisons and more speed in running time. The time complexity of this algorithm for scanning all records in database is *O (WT)* where W is the priority queue size and T is the total number of records in database. Figure 4 shows the priority queue algorithm for scanning the bibliographical dataset.

```
Input:  Sorted Bibliography Dataset
        with Created Keys
Output: Possible Duplicate Record
        with Similarity Values
Begin
1. Read through sorted dataset sequentially
2. while ( Sorted dataset hasNext() )
Begin
    3. Retrieve one record ($R_j$) from sorted dataset
    4. Test record ($R_j$) contains in Priority Queue
    5. If ( $R_j$ not contains in Priority Queue)
    Begin
      6. for (all records in Priority Queue)
        Begin
        7. Compute similarity values between $R_j$
           and other records in Priority Queue
           using Smith-Waterman algorithm
        8. If ( Similarity-Value >= Threshold-Value)
           Store the record pairs in duplicate table
           with their similarity scores
        End
      9. If no match found
        Begin
          10. if (PriorityQueue.Size() >10)
          Begin
            11. Remove record with lowest priority
                in Priority Queue
            12. Push $R_j$ into Priority Queue
          End
          13. Else Push $R_j$ into Priority Queue
        End
    End
    14. Else $R_j$ contains in Priority Queue
          Read next record $R_{j+1}$
  End
End
```

Figure 4. Priority Queue Algorithm

3.2.3.2. Matching Criteria

Bibliographical records from DBLP are compared across a set of five matching criteria and we refer to them as the similarity of corresponding fields. Most of the data fields in a bibliographical record are the free-text strings. We denote the similarity functions for each field by using Smith-Waterman algorithm respectively as shown in the following.

1. Smith-Waterman(Key) =String edit distance of Cr_key field using Smith-Waterman algorithm
2. Smith-Waterman(Author) =String edit distance of author-name field using Smith-Waterman algorithm
3. Smith-Waterman(Title) =String edit distance of Title field using Smith-Waterman algorithm
4. Smith-Waterman(Proceeding) =String edit distance of Proceeding field using Smith-Waterman algorithm
5. Boolean (Year) =Boolean matching as similarity function (1 or 0) of year field

3.2.3.3. Smith-Waterman Algorithm

To measure the similarity value between records we use Smith-Waterman algorithm in this system. Given two strings of characters, it uses dynamic programming to find the lowest cost series of changes that converts one string into the other, i.e. the minimum "edit distance" weighted by cost between the strings [8]. Costs for individual changes, which are mutations, insertion, or deletions, are parameters of the algorithm.

Much of the power of the Smith-Waterman algorithm is due to its ability to introduce gaps in the records. A gap is a sequence of non-matching symbols; these are seen as dashes. It works by computing a score matrix H. One of the strings is placed along the horizontal axis of the matrix, while the second string goes along the vertical axis. An entry H (i,j) in this matrix is the best possible matching score between the prefix 1……i of the second string. When the prefixes (or the entire strings) match exactly, then the optimal alignment can be found along the main diagonal. For approximate matches, the optimal alignment is within a small distance of the diagonal. A matrix H is built as follows:

$H(i, 0) = 0, 0 \leq i \leq m$
$H(0, j) = 0, 0 \leq j \leq n$

$$H(i,j) = \max \begin{cases} 0 \\ H(i-1, j-1) + w(a_i, b_j) \\ \text{Match or Mismatch} \\ H(i-1, j) + w(a_i, -) \text{ Deletion} \\ H(i, j-1) + w(-, b_j) \text{ Insertion} \end{cases}$$

Where:

- a, b = Strings over the Alphabet ∑
- m = length(a)
- n = length(b)
- H(i, j) – is the maximum Similarity-Score between a suffix of a[1…i] and a suffix of b[1…j]
- w(c, d) , c, d Є ∑ ∪ {'-'}, '-' is the gap scoring scheme

Sequence 1=ACACACTA
Sequence 2=AGCACACA

Table 3. Example of Match Scores Matrix Produced by Smith Waterman Algorithm

	-	A	C	A	C	A	C	T	A
-	0	0	0	0	0	0	0	0	0
A	0	2	1	2	1	2	1	0	2
G	0	1	1	1	1	1	1	0	1
C	0	0	3	2	3	2	3	2	1
A	0	2	2	5	4	5	4	3	4
C	0	1	4	4	7	6	7	6	5
A	0	2	3	6	6	9	8	7	8
C	0	1	4	5	8	8	11	10	9
A	0	2	3	6	6	10	10	10	12

To obtain the optimum local alignment, we start with the highest value in the matrix (i, j). Then, we go to the biggest value among those in position (i-1, j), (i, j-1), and (i-1, j-1). We keep the process until we reach a matrix cell with zero value, or the value in position (0, 0). Once we've finished, starting with the last value, we reach (i, j) using the previously-calculated path. A diagonal jump implies there is an alignment (either a match or a mismatch). A top-down jump implies there is a deletion. A left-right jump implies there is an insertion.

For Table 3, we get:

Sequence 1=A-CACACTA
Sequence 2=AGCACAC-A

All experiments in this system use the Smith-Waterman algorithm with exact match scores +2 and no match score -1.The final score calculated by the algorithm is normalized to range between 0.0 and 1.0 by dividing by 2 times the length of the smaller of the two records being compared. Example of similarity values using Smith-Waterman algorithm is shown in Table 4.

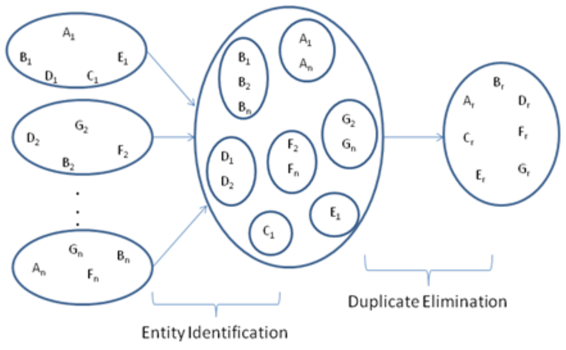
Table 4. Example results for Similarity Values

ID1	ID2	key	author	title	Proceed -ing	year
2	3	1.00	0.88	0.75	0.70	1
2	73	0.42	0.65	0.50	0.30	0
5	54	0.65	0.70	0.65	0.76	1
56	123	1.00	0.89	0.80	0.76	0

The time complexity of this algorithm for comparing two sequences is *O (mn)*, where *m* and *n* are the lengths of the two sequences being compared.

3.3. Duplicate Elimination

During the elimination process, only one record of duplicates is retained and eliminated other duplicate records [1, 3]. The elimination process is very important to produce a clean data.



A_1 B_1 E_1 D_1 C_1
G_2 D_2 F_2 B_2
G_n B_n A_n F_n
B_1 B_2 B_n A_1 A_n G_2 G_n D_1 D_2 F_2 F_n C_1 E_1
B_r A_r D_r C_r F_r E_r G_r
Duplicate Elimination
Entity Identification

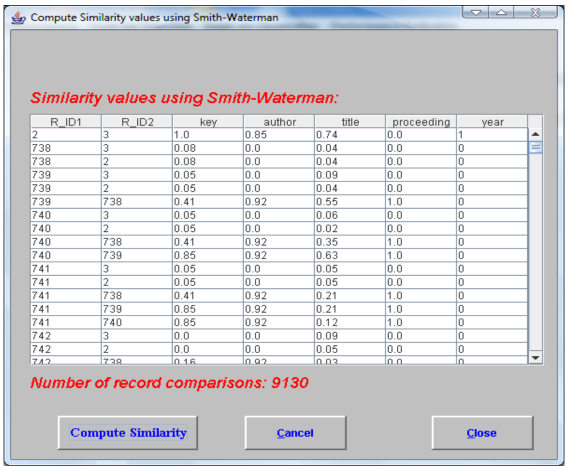
Figure 5. Duplicate elimination process

It can be consider a two-step process as illustrated in Figure 5. Entity identification step is discussed in Section 3.2. The elimination step use the duplicate record pairs identify during duplicate detection as an input. Then, all duplicate records are grouped in arraylist. Then only one record is retained and others are deleted.

4. System Implementation

In our proposed system, we use bibliographical XML dataset from DBLP database which contains 860 records. Firstly, we perform parsing and standardizing bibliography XML dataset for preprocessing phase. In duplicate detection phase, key creation process must be performed by extracting relevant or portions of author names and year fields.

Next, the preprocessed records are sorted according to keys that get from key creation process. These sorted dataset are input to priority queue algorithm. Then, we compute similarity value using smith-waterman algorithm. Figure 6 shows the result of similarity values using smith-waterman algorithm. Figure 7 shows the result after duplicate detection. Then, only one record with maximum year is retained and other duplicate records are deleted.

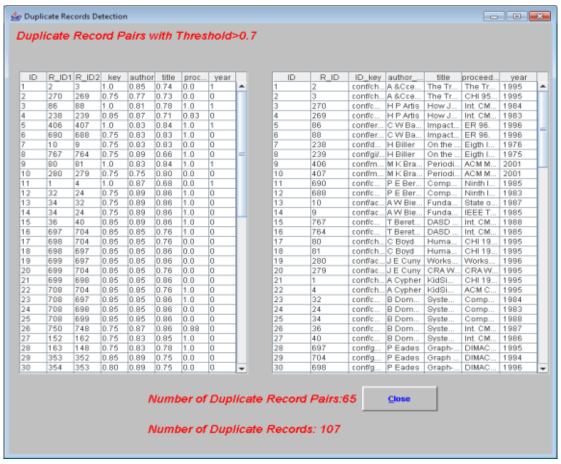


Similarity values using Smith-Waterman:

R_ID1	R_ID2	key	author	title	proceeding	year
2	3	1.0	0.85	0.74	0.0	1
738	3	0.08	0.0	0.04	0.0	0
738	2	0.08	0.0	0.04	0.0	0
739	3	0.05	0.0	0.09	0.0	0
739	2	0.05	0.0	0.04	0.0	0
739	738	0.41	0.92	0.55	1.0	0
740	3	0.05	0.0	0.06	0.0	0
740	2	0.05	0.0	0.02	0.0	0
740	738	0.41	0.92	0.35	1.0	0
740	739	0.85	0.92	0.63	1.0	0
741	3	0.05	0.0	0.05	0.0	0
741	2	0.05	0.0	0.05	0.0	0
741	738	0.41	0.92	0.21	1.0	0
741	739	0.85	0.92	0.21	1.0	0
741	740	0.85	0.92	0.12	1.0	0
742	3	0.0	0.0	0.09	0.0	0
742	2	0.0	0.0	0.05	0.0	0
742	738	0.16	0.92	0.03	0.0	0

Number of record comparisons: 9130

Figure 6. Result of Similarity Values using Smith-Waterman Algorithm



Duplicate Record Pairs with Threshold>0.7

ID	R_ID1	R_ID2	key	author	title	proc...	year
1	2	3	1.0	0.85	0.74	0.0	1
2	270	269	0.75	0.77	0.73	0.0	0
3	86	88	1.0	0.81	0.78	1.0	1
4	238	239	0.85	0.87	0.71	0.83	0
5	406	407	1.0	0.83	0.84	1.0	1
6	690	688	0.75	0.83	0.83	1.0	0
7	10	9	0.75	0.83	0.83	0.0	0
8	767	764	0.75	0.89	0.66	1.0	0
9	80	81	1.0	0.83	0.84	1.0	1
10	280	279	0.75	0.75	0.80	0.0	0
11	1	4	1.0	0.87	0.68	0.0	1
12	32	24	0.75	0.89	0.86	1.0	0
13	34	32	0.75	0.89	0.86	1.0	0
14	34	24	0.75	0.89	0.86	1.0	0
15	36	40	0.85	0.89	0.86	1.0	0
16	697	704	0.85	0.85	0.76	1.0	0
17	698	704	0.85	0.85	0.76	0.0	0
18	698	697	0.85	0.85	0.86	0.0	0
19	699	697	0.85	0.85	0.86	0.0	0
20	699	704	0.85	0.85	0.76	0.0	0
21	699	698	0.85	0.85	0.86	0.0	0
22	708	704	0.85	0.85	0.76	1.0	0
23	708	697	0.85	0.85	0.86	1.0	0
24	708	698	0.85	0.85	0.86	0.0	0
25	708	699	0.85	0.85	0.86	0.0	0
26	750	748	0.75	0.87	0.86	0.88	0
27	152	162	0.75	0.83	0.85	1.0	0
28	163	148	0.75	0.83	0.78	1.0	0
29	353	352	0.85	0.89	0.75	0.0	0
30	354	353	0.80	0.89	0.75	0.0	0

ID	R_ID	ID_key	author_...	title	proceed...	year
1	2	conf/ch...	A &Cce...	The Tr...	The Tr...	1995
2	3	conf/ch...	A &Cce...	The Tr...	CHI 95...	1995
3	270	conf/c...	H P Artis	How J...	Int. CM...	1984
4	269	conf/c...	H P Artis	How J...	Int. CM...	1983
5	86	conf/er...	C W Ba...	Impact...	ER 96.	1996
6	88	conf/er...	C W Ba...	Impact...	ER 96.	1996
7	238	conf/d...	H Biller	On the ...	Eigth I...	1976
8	239	conf/gi/...	H Biller	On the ...	Eigth I...	1975
9	406	conf/m...	M K Bra...	Periodi...	ACM M...	2001
10	407	conf/m...	M K Bra...	Periodi...	ACM M...	2001
11	690	conf/c...	P E Ber...	Comp...	Ninth I...	1985
12	688	conf/c...	P E Ber...	Comp...	Ninth I...	1983
13	10	conf/ac...	A W Bie...	Funda...	State o...	1987
14	9	conf/ac...	A W Bie...	Funda...	IEEE T...	1985
15	767	conf/c...	T Beret...	DASD ...	Int. CM...	1988
16	764	conf/c...	T Beret...	DASD ...	Int. CM...	1985
17	80	conf/ch...	C Boyd	Huma...	CHI 19...	1995
18	81	conf/ch...	C Boyd	Huma...	CHI 19...	1995
19	280	conf/ac...	J E Cuny	Works...	Works...	1996
20	279	conf/ac...	J E Cuny	CRA W...	CRA W...	1995
21	1	conf/ch...	A Cypher	KidSi...	CHI 19...	1995
22	4	conf/ch...	A Cypher	KidSi...	ACM C...	1995
23	32	conf/c...	B Dom...	Syste...	Comp...	1984
24	24	conf/c...	B Dom...	Syste...	Comp...	1983
25	34	conf/c...	B Dom...	Syste...	Comp...	1988
26	36	conf/c...	B Dom...	Syste...	Int. CM...	1987
27	40	conf/c...	B Dom...	Syste...	Int. CM...	1986
28	697	conf/g...	P Eades	Graph-...	DIMAC...	1995
29	704	conf/g...	P Eades	Graph ...	DIMAC...	1994
30	698	conf/g...	P Eades	Graph-...	DIMAC...	1996

Number of Duplicate Record Pairs:65

Number of Duplicate Records: 107

Figure 7. Result of Duplicate Detection

5. Performance Evaluation

The performance evaluation shown in Figure 8 is evaluated according to the following percentages:

(1) Recall
(2) False Positive Error
(3) False Negative Error
(4) Precision

(1) Recall

It is identified as the percentage of duplicate records being correctly identified by the system.

$$Recall = \frac{no\ of\ identified\ duplicates}{no\ of\ actual\ duplicates} * 100\ \%$$

(2) False Positive Error

This is the percentage of records wrongly identified as duplicates.

$$FP = \frac{no\ of\ wrongly\ identified\ duplicates}{total\ no\ of\ identified\ duplicates} * 100$$

(3) False Negative Error

It is the percentage of duplicate records that are not detected by system.

$$FN = 100\% - Recall$$

(4) Precision

It is the percentage of the information reported as relevant by system that is correct.

$$Precision = 100\% - FP$$

The higher the percentage of precision and recall, the better the result can get. Also, the less in the percentage of FP and FN, the better the result can get.

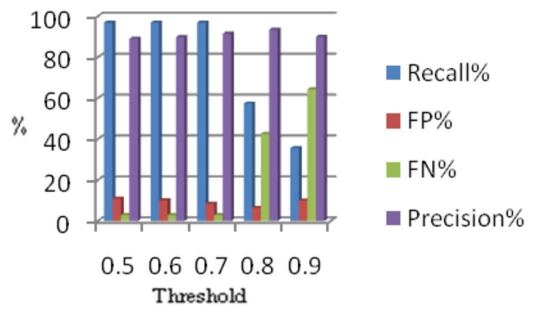


Figure 8. Performance Result

In this system, the threshold value 0.5, 0.6 and 0.7 are the best result for duplicate detection because there are less in the percentage of FP and FN and high percentage of precision and recall. Figure 8 shows the performance evaluation of duplicate detection.

6. Conclusion

Our proposed framework is designed to clean duplicate data for improving data quality. This work focuses on the identifying the duplicate records in bibliographical XML dataset from DBLP repository using priority queue algorithm and eliminating them to produce clean data. The experiments resulted in high duplicate detection accuracy while significantly performing many fewer record comparisons.

Smith-Waterman algorithm is implemented to measure the similarity value. The main purpose of the algorithm is to determine similar values between two records and to optimize the similarity measure. It is performing local sequence alignment and so it compares segments of all possible lengths rather than looking at the total sequence. So it is higher similarity measure than other edit distance algorithms.

Time is critical in data cleaning large database. In this system, efficient priority queue method for detection is used to reduce the time taken on each comparison. Efficient duplicate detection and elimination approach is developed to obtain good result of duplicate detection and elimination by reducing false positives. Performance evaluation shows that there was significant time saving and improved duplicate results. The system is mainly developed to increase the speed of the duplicate data detection and elimination process and to increase the quality of the data by identifying true duplicates and strict enough to keep out false-positive.

7. References

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti, *Eliminating Fuzzy Duplicates in Data Warehouses*. VLDB, pages 586-597, 2002.

[2] C. Batini, M. Lenzerini, and S. Navathe. A comparative analysis of methodologies for database schema integration.*ACM Computing Surveys*, 18(4):323-364(1986).

[3] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. *Duplicate Record Detection: A Survey*, IEEE TKDE, 19(1):1-16, 2007.

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

[5] M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *Proceeding of the ACM SIGMOD International Conference on Management of Data*, pp. 127-138(1995).

[6] W. Kim, I. Choi, S. Gala, and M. Scheevel. On resolving schematic heterogeneity in multidatabase systems. *Distributed and Paralle Database*, 1(3): 251-279(1993).

[7] Larry P. English, J.: Column “Plain English on Data Quantity”, DM Review: http://www.dmreview.com, last accessed 02/10/99.

[8] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195-197(1981).

[9] Htike. Thin Thin, “An Association Rule Based Approach to Duplicate Detection in Bibliographical Records.” In *Proceedings of the 3rd International Conference on Computer Application* (ICCNA), 2005.