International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Techniques for Duplicate Detection in Hierarchical Data

Suvarna Kale [1], Basha Vankudothu [2]

[1]Department of Computer Engineering, GSMCOE,, Balewadi, Savitribai Phule University, Pune, India

Abstract: ***Duplicate detection is nothing but finding multiple representations of a same object and also object which are represented in a dataset. The duplicate detection is important to integration and data cleaning applications and it is studied for relational data in single table, but now data is stored in complex form. In this paper we improve the efficiency and effectiveness of duplicate detection by considering relationship between ancestors and descendants. We apply this strategy by implementing two algorithms RECONA and ADAMA. Recona re-examine an object if its induce neighbours is duplicates. This will reduce re-comparison of elements. Adama is efficient because it does not allow re-comparison***

Keywords: Duplicate detection, XML data, hierarchical structure, candidate pair

1. Introduction

Duplicate detection is useful in data cleansing ,data integration [6], personal information management .Uptil now Duplicate detection is studied for relational data which is stored in a single table. But there are hierarchical data which is nothing but distinct set of data items that are related to each other by hierarchical relationships which is present in more complex form.

The area where for duplicate detection is mostly used is customer relationship management (CRM) where there can be multiple representation of the same entity. XML is mostly popular on the Web for data publish process and in the organizations for exchange of data.

The current work in XML world has shown that the efficiency and effectiveness of duplicate detection has improved by considering relationships between ancestors and descendants. In Hierarchical data, one data item is the parent of another item. So this Hierarchical relationalship is represented using parent and child relationships . In this each parent can have many children, but each child has only one parent. so we cannot use conventional approach. Hierarchical data is mostly used for exchange of information on web and in many other places.

XML document is represented as a tree like structure. Several problems occur while doing data integration from different data sources such as when distributed and heterogeneous data source is combined. When we combine data from different data sources, the ideal result should be a unique complete and correct representation for every object to achieve data quality. So this ensures that only one representation of object is present in the database.

In this paper, we find a duplicate detection approach for XML data, which uses all kinds of relationships between entities,
i.e., 1:1, 1:n, and n:m. The basic idea is presented in [5]. The algorithms [7] use pairwise comparisons more than once to increase effectiveness . The focus of this paper is on efficient implementation of dependencies between entities. We use two algorithms RECONA and ADAMA. Recona is used for improving efficiency and Adama is used for improving effectiveness. We are also going to compare RECONA and ADAMA with the state of art XMLDUP approach with efficiency and Effectiveness evaluation. In XMLDUP approach , it uses the Bayesian Network model for finding duplicates. It uses this model to compute similarity between XML object, and according to this similarity it considers whether this XML objects are duplicates or not. It also uses network pruning algorithm to improve Bayesian Network evaluation time.

2. Related Work

In this section we have discussed various duplicate detection algorithms and techniques used earlier. Research in duplicate detection comes under two categories ,These are the techniques for improving efficiency and effectiveness. Efficiency deals with improvement in precision and recall.

Delphi[1] uses a top down approach which considers immediate attributes of objects and also their children and parents in a complex datawarehouse.but the main drawback here is it don't compares al pairs of tuples in the hierarchy because it evaluates the outermost layer first and then proceeds to the innermost layer of the xml.

Lus Leitao, Pavel Calado, and Melanie Herschel suggested a method XMLDUP for XML duplicate detection. Bayesian Network is used in XMLDup to determine whether probability of two XML elements being duplicates or not. Network Pruning Strategy is used To improve the efficiency of network evaluation. XMLDup showed better results better results with respect to both efficiency and Effectiveness when compared to another method [2]

SXNM (Sorted XML Neighborhood Method) is a method proposed by S. Puhlmann is a duplicate detection method which contains relational sorted neighborhood approach (SNM) to XML data. Just like the original SNM, the idea behind is to avoid performing useless comparisons between objects by grouping together those that are more likely to be similar [3].

Edit distance is used for measuring of Detecting the duplicates between XML entities which involves detecting similarity between entities [4].

M. Weis et.al has proposed Dogmatix framework. It consists of three main steps: candidate definition structure, duplicate definition and duplicate detection. Dogmatix method compares XML elements on the similarity of their parents, children and structure [6].

3. Proposed Work

In the previous work , there was Bayesian Network Construction which considers not only information within elements but also the way that how information is Structured . But this structure does not consider ordering of elements.

In this paper we are going to use two algorithms RECONA and ADAMA which considers ordering strategy. It finds the duplicate detection by using ascending order of r and using the relationship between objects , where comparison order is obtained by computing a rank r(v,v') for every candidate pair (v,v'). Whenever there is increase in similarity, RECONA recomputes similarity between two objects. ADAMA avoids re-comparison to increase efficiency. For comparison order ,if we choose ascending order of r ,we always choose to compare objects first that have fewer duplicates. In this case ,the ripple effect to neighbours is low if two objects are found duplicates. So RECONA avoids recomparison if we carefully choose comparison order which improves efficiency. Careful chosen order of ADAMA improves effectiveness.

RECONA Algorithm

The RECONA algorithm is the perfect algorithm for finding duplicate detection. The RECONA algorithm has two phases. The first is initialization phase and the other is comparison phase. Detecting duplicates to an object is based on the assumption that it may affect similarity and duplicate classification on other object.

The initialization phase (lines 2-10) contains all pairs of candidates which is defined in a priority queue *OPEN* which defines ascending order of rank r. DUPS is a set of duplicate pairs which contains duplicates pairs to avoid unnecessary re-comparison

```
1 Procedure ReconA()
2 G: data Graph;
3 OPEN: priority queue of candidate pairs
4 ordered in ascending order of r ;
5 DUPS: set of duplicate pairs;
6 CLOSED: set of possibly re-classified pairs;
7 θ : similarity threshold;
8 Initialize G;
9 Add all candidate pairs to OPEN;
10 while OPEN not empty do
11 begin
12 (v_i , v_j) ← OPEN.popFirst();
13 sim = sim(v_i , v_j);
14 if sim > θ then
15 begin
16 DUPS := DUPS ∪ {(v_i , v_j)};
17 updateOpenReconA(v_i , v_j);
18 end
19 end
```

Listing 1 RECONA Algorithm

```
1 procedure updateOpenReconA(Vertex v, Vertex v')
2 D(v, v') = {(n1, n2)|n1 ∈ D(v) ∧ n2 ∈ D(v') ∧ n1 ≠ n2};
3 for all (n1, n2) ∈ D(v, v') do
4 if (n1, n2) not ∈ DUPS then
5 begin
6 r_update := r(n1, n2);
7 if (n1, n2) ∈ OPEN then
8 OPEN.updateRank((n1,n 2), r_update );
9 else if (n1, n2) ∈ CLOSED then
10 OPEN.Push((n1, n2), r_update );
11 end
```

Listing 2: Updating OPEN in RECONA

ADAMA Algorithm

ADAMA works similar to RECONA ,but the important difference is that , once candidate pair is classified, we do not add it to open regardless of whether they are classified as duplicates or not . So pairwise comparisons are not performed more than once.

We define another set of candidates called NONDUPS in the initialization phase. It avoids re-comparison and compute Rank r in which set of neighbour pairs is indicated as neighbor pairs not in DUPS and not in OPEN. In ADAMA's algorithm , if similarity value of pairs is below threshold , it is added to NONDUPS and is never taken for re-comparisons which is defined in updateOpenAdamA. In updateOpenAdamA procedure we update the ranks of pairs that are present in OPEN. So the complexity of ADAMA algorithm is N because we do not allow re-comparison of pairs.

```
1 procedure AdamA()
2 G, OPEN, t, sim, DUPS as in ReconA;
3 NONDUPS: set of non-duplicate pairs;
4 Initialize G;
5 Add all candidate pairs to OPEN;
6 while OPEN not empty do
7 begin
8 (vi, vj ) ← OPEN.popFirst();
9 sim = sim (vi, vj );
10 if sim > θ then
11 updateOpenAdamA(vi,vj);
12 else
13 NONDUPS := NONDUPS ∪ {(v1, vj )};
14 end
```

Listing 3: ADAMA Algorithm

```
1 procedure updateOpenAdamA(Vertex v, Vertex v' )
2 D(v, v' ) = {(n1, n2)|n1 ∈ D(v) ∧ n2 ∈ D(v' ) ∧
 n1 ≠ n2};
3 forall (n1, n2) ∈ D(v, v' ) do
```

```
4 if (n1, n2) not ∈ DUPS ∪ NONDUPS then
5 begin
6 r_update := r(n1, n2);
7 if (n1, n2) ∈ OPEN then
8 OPEN.updateRank((n1,n 2), r_update ;
9 end
```

Listing 4: Updating OPEN in ADAMA

4. Experiments on Duplicate Detection

In this section we will present an evaluation of the Recona & Adama algorithm described in the previous sections. We have evaluated the algorithm in terms of effectiveness and efficiency. First, we evaluate effectiveness by comparing it to a duplicate detection system, called XMLDUP , that is most competitive. We then evaluate the efficiency of Recona & Adama algorithm. Testing the impact of data quality on duplicate detection is important to confirm the effectiveness of a given algorithm.

4.1 Effectiveness Evaluation

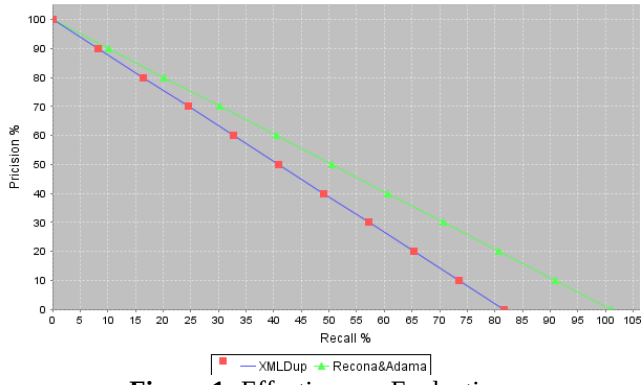

Figure 1: Effectiveness Evaluation

Precision and recall are the basic measures used in evaluating search strategies.The Graph shows that Recona & Adama are more effective than the previous duplicate detection strategy XMLDUP.

4.2 Efficiency Evaluation

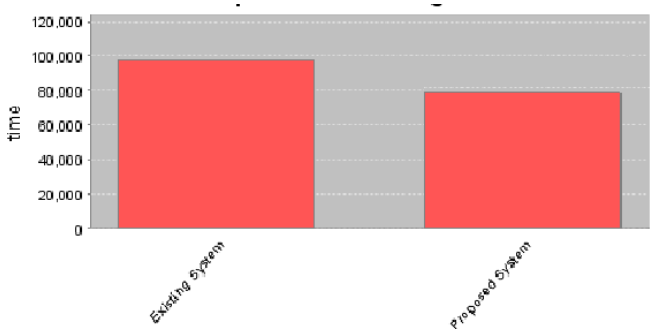

Figure 2: Efficiency Evaluation

In the above fig we did efficiency evaluation by comparing Existing System XMLDUP with the Proposed system Recona & Adama. It shows that Existing System takes more time for evaluation than Proposed system

5. Conclusion

Duplicate detection is done for finding different representation of the same real world object which is called duplicate. These duplicate data are not exactly equal due to errors in the data. Due to this duplicate detection is challenging task in data cleaning and data integration processes. The efficiency of the duplicate detection method is improved by using a novel duplicate detection approach for XML data which performs good in all kinds of relationships between entities i.e. 1:1, 1:n and m:n. The strategy for comparisons we used here is pairwise comparison in ascending order of rank. This strategy applied is RECONA & ADAMA algorithm. RECONA algorithm also called re-examining algorithm does pairwise comparison which is performed more than once. So the proposed comparison order reduces number of re-comparisons. In ADAMA , recomparisons are avoided to increase efficiency. In Experiments , The number of re-comparisons for RECONA gets reduced through the order obtained using ascending rank r for a high interdependency between entities. For ADAMA , In recall and precision the order using r performs slightly better than other orders for high interdependency.

6. Acknowledgment

I would like to express my gratitude to Prof. Ratnaraj Kumar, Head of the Department and my project Guide for providing me with adequate facilities to complete this Paper. I express my gratitude to him for his support and suggestions. I thank Prof.Basha V., my project guide for his cooperation and helps.

References

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *International Conference on Very Large Databases (VLDB)*, Hong Kong, China, 2002.
[2] Lus Leitao, Pavel Calado, and Melanie Hersche, "Efficient and Effective Duplicate Detection in Hierarchical Data," IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,VOL. 25, NO. 5, MAY 2013
[3] S. Puhlmann, M. Weis "XML Duplicate Detection Using Sorted Neighborhoods," Proc. Conf. Extending Database Technology (EDBT), pp. 773-791, 2006.
[4] Melanie Weis and Felix "Detecting duplicate objects in XML documents", Humboldt-Universität zu Berlin, Germany, Workshop on Information Quality in Information Systems, 2004
[5] M. Weis and F. Naumann. Detecting duplicates in complex XML data. In *International Conference on Data Engineering (ICDE)*, Atlanta, Georgia, 2006.
[6] M. Weis and F. Naumann, "Dogmatic Tracks Down Duplicates in XML," Proc. ACM SIGMOD Conf. Management of Data, 2005.
[7] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *International Conference on the Management of Data (SIGMOD)*, Baltimore, MD, 2005.