RULE BASED METHOD FOR ENTITY RESOLUTION USING DISTINCT TREE CONSTRUCTION

Ammu Archa.P
Student, CSE department,
SCTCE,
Thiruvananthapuram, India.
ammuarcha92@gmail.com

Lekshmy.D.Kumar
Assistant Professor, CSE department,
SCTCE,
Thiruvananthapuram, India.
lechuvinan@hotmail.com

Abstract— **Entity resolution is the process of identifying the records that refer to the same entity. Rule based ER works by generating rules from the training dataset obtained from the given dataset and applying these rules to the records in the dataset. This method is very time consuming and tedious as the size of the rule set generated is very large. Also, the rules generated are not efficient enough to classify the records correctly. So a distinct tree construct is proposed to generate the rules from the dataset. Distinct tree is constructed by arranging the dataset in a particular order before rule generation step. Experiments shows that the accuracy of rules generated using distinct tree method is more accurate and fast than simple Rule based ER.**

Index terms—**Entity, Entity resolution, Rules, Length threshold, Distinct tree.**

I. INTRODUCTION

Relational database is a collection of entities where each entity is an object which has physical or logical existence. Each tuple in the database is an entity while each entity contains more than one tuple. It is important to identify the records referring to the same entity since it has many real life applications. Applications include comparative shopping, linking census records, public health, web search etc. The process of identifying the records referring to the same entity is called Entity Resolution(ER).

ER is one of the most important processes in data mining and arises in many real life applications like information integration and information retrieval. Because of its importance there are many methods to perform ER. Fellegi and Sunter has proposed a probabilistic model in [2] using bayes decision rule for ER. In bayes decision rule the record pair will be classified into match or unmatch class based on probability. If the probability of match class is greater than the probability of unmatch class then the record pair will be in the match class and vice versa. The disadvantage is that probabilities should be known in advance.

In the work done by cochinwala et.al [3] used the well known CART algorithm [4] which comes under supervised learning technique. Here, a binary decision tree is constructed and is trained using training samples. The disadvantage is that a large number of training sample are required.

Sarwagi and bamidipati [5] designed ALIAS, an learning based technique. This is similar to supervised learning technique but the difference is that here user can give feedback. If the user is not satisfied with the classification, he/she can change the training sample and the process continues until the user is happy with the result. Lingli, jianzhong and hong proposed a rule based method for entity resolution [1]. In this method rules are used to perform entity resolution or classification.

The remainder of the paper is organized as follows: An overview of related work is given in section I. Section II gives idea of the existing system. Section III describes the proposed system. The two systems are evaluated experimentally in Section IV and the paper is concluded in Section V.

II. EXISTING SYSTEM

In existing system, rules are generated from the dataset given and these rules are used for entity identification. Before generating the rules, entity set is generated from the given dataset. From the entity set training set is generated. Rules are then generated from these training set. Rules are generated for each entity set. Each attribute-value pair in the training set is considered as a rule. Rules generated should satisfy length requirement. Length requirement states that the length of the rule generated should always be less than or equal to the given length threshold i.e. $|r| <= l$ where l is the length threshold. Length threshold can be any value that the user selects but it should not be very high or very low. In the existing system length threshold is fixed as 2.

After generating the rule, rule coverage is checked. Coverage is the number of records a particular rule can satisfy. Based on the coverage rule is classified as basic rule or preliminary rule. Basic rule is said to be valid and preliminary rule is said to be invalid. If a rule covers records in only one entity then that rule is said to be a basic rule and if a rule covers records in more than one entity then that rule is classified as preliminary rule. If a rule is basic then it is stored

in R and if it is preliminary it is inserted in L^x & L^y. After generating all the basic rules, rules in L^x and L^y are considered. Each rule in L^x is combined with rule in L^y ad the coverage is checked. If it is a valid rule then it is added to R else it is added to L^y. This process of rule generation is continued until the length threshold is met.

After all the basic rules are identified, the tuples in the training set that cannot be identified using the rules generated are identified. For such tuples the entire attribute-value pair in the tuple is considered as a rule. For such tuples the rule length criteria is neglected. There may be cases where more than one rule identifies same record. In such cases one among many rules need to be selected to identify that record. So in such cases rule with minimum length is chosen as the rule for that record.

Now, rules for the entire training set have been discovered. Next step is to perform entity resolution on the entire dataset using these rules. In entity resolution, each record is scanned and the entity to which it refer to is identified. For that each record is taken and finds all the rules satisfied by the record. The idea is to compare record with each of the generated rules. Now rule set of record is compared with rule set of each of the entities. Assumption is that the record belongs to that entity having maximum common rule with the rule set of the record.

Here, no special criteria are used for rule generation. Simply each attribute-value pair in the training set are considered as rule. As a result, during entity resolution step some records are misclassified and some are unclassified. Also, the rule generation is time consuming and the number of rules generated is large. A more efficient and accurate method for rule generation is proposed to avoid the disadvantages of the existing system.

III. PROPOSED SYSTEM

In the proposed system all the steps are similar to that of the simple Rule based entity resolution other than the rule discovery step. Architecture of the proposed system is given in Fig.1. First the entity set is created and then training set is created. Then rules are generated using a distinct tree construction method and finally entity resolution is performed.

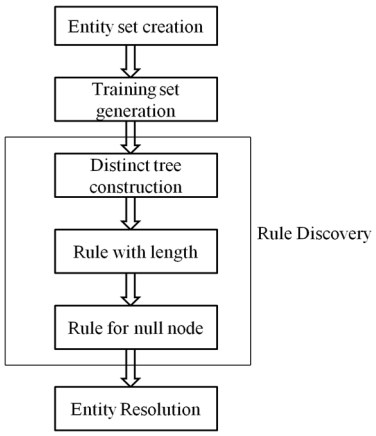

Fig.1. Architecture of Proposed system

Algorithm 1 presents the algorithm for rule discovery.

Algorithm 1 Rule Generation

Input: length threshold l, Training set $T = \{T_1, T_2 \ldots..T_n\}$
Output: Rule set R

1: $T \leftarrow \{T_1, T_2 \ldots\ldots..T_n\}$
2: $T_{Tree} \leftarrow$ DistTreeGen(T)
3: R_{Len} , $N \leftarrow$ RL (l, T_{Tree})
4: for each n_i in N do
5: $R_{RNN} \leftarrow$ RNN (n_i)
6: end for
7: $R \leftarrow R_{Len} \cup R_{RNN}$
8: Return R

Given the length threshold l and the training set T, distinct tree is constructed first by DistTreeGen procedure (line 2), denoted by T_{Tree}.

Rules for entity resolution that satisfies the length requirement are found by RL procedure (line 3), denoted by R_{Len} and rules that cannot be formed by using RL algorithm is found using RNN algorithm (line 5), denoted by R_{RNN}. RNN procedure does not consider length criteria. Here the nodes of the tree are stored in N. Finally the rule set R is the union of rules in R_{Len} and R_{RNN}.
DistTreeGen. Given the training set T DistTreeGen (shown in algorithm 2) generates the distinct tree.

Algorithm 2 DistTreeGen

Input: Training set $T = \{T_1, T_2 \ldots\ldots..T_n\}$
Output: T_{Tree}

1: Initialize
2: Sel_attr=Ø
3: for each r_i in T_i
4: records $\leftarrow r_i$
5: end for
6: $T_{Treei} \leftarrow$Perform Group (records, Sel_attr)
7: Return $T_{Tree} = \{T_{Tree1} \cup T_{Tree2} \cup \ldots \cup T_{Treen}\}$

8: procedure Perform Group (records, Sel_attr)
9: if (Sel_attr < no: of attributes) then
10: Best_attr, Values$\leftarrow$GetBestAttr (records)
11: for each value in Values do
12: $n_i \leftarrow$ NodeGen(Best_attr,value)
13: $T_{Treei} \leftarrow n_i$
14: PerformGroup(records,Sel_attr)
15: end for
16: end if
17: return T_{Treei}
18: end procedure

```
19: Procedure NodeGen(Best_attr,value)
20: Sel_attr = Best_attr + Sel_attr
21: ni ← (Best_attr,value)
22: for each ri in records
23:  if (Best_attr value of(ri) ==value) then
24:    records ← ri
25:  end if
26: end for
27: return ni
28: end procedure
```

Distinct tree is constructed for each entity set using their attribute and value. PerformGroup procedure (line 6) is used to generate distinct tree for each records, denoted by T_{Treei}. Procedure PerformGroup (lines 8-18) is recursively called until the entire attribute is considered.

GetBestAttr in PerformGroup (line 10) is used to find the attribute with minimum number of distinct value. The result of GetBestAttr is the attribute and its value and is stored in Best_attr and values respectively. NodeGen in PerformGroup (line 12) is used to generate the nodes for the distinct tree.

NodeGen procedure (line 19-28) generates the node for distinct tree using Best_attr and value. The attribute with least number of distinct values is considered as the root node or parent node of the tree. The child node will be the attribute having next minimum attribute values. After the tree construction next step is to generate rules from the tree.

As in the case of simple rule based ER, here also rules that satisfies the length criteria as well as rules that does not satisfies length criteria are considered. Algorithm 3 presents the algorithm for rule generation that satisfies length threshold, denoted as RL.

Algorithm 3 RL

Input: l, T_{Tree}
Output: N, R_{Len}

```
1: Initialize
2: for each ni in TTreei
3:   re = GetParent(ni)
4:   if (re! = Ø) then
5:     ni . rule = re
6:     else
7:     re = GetRule(ni)
8:      if (|re| < = l & re is valid) then
9:      ni . rule = re
10:    else
11:    ni . rule = Ø
12:    Ni← ni
13:   end if
14:  end if
15: end for
16: Return N = {N1U N2U ……U Nn}
17: Return RLen={TTree1.rule U TTree2.rule U…UTTreen.rule}

18: Procedure GetParent(ni)
19: while (ni != Ø)
20:   if (ni.rule!= Ø) then
21:      return ni.rules
22:      ni = parent of ni
23:   end if
24: end while
25: return null
26 end procedure

27: Procedure Get Rule (ni)
28: re = Ø
29: while (parent of ni != Ø)
30:     r = ni
31:     if (re == Ø) then
32:        re = r
33:       else
34:       re = re ∧ r
35:    end if
36:  ni = parent of ni
37: end while
38: return re
39: end procedure
```

Given the distinct tree as input, RL algorithm will generate rules that satisfy the length threshold. Each node in the tree is considered and we check whether rule is assigned for the parent node of the current node under consideration using GetParent procedure (line 3). If GetParent procedure returns null then rule is to be generated for the current node. GetRule procedure (line 7) is used to assign rule for the node whose parent node has null rule. Else rule of the parent node is assigned as the rule for the current node under consideration. Finally we check whether rule generated satisfies the length threshold and is valid (line 8-12). If true then the rule is assigned to the node otherwise null value is assigned to the node.

GetParent procedure (line 18-26) is used to retrieve the rule of the parent node. If the rule of the parent node is null, then the procedure returns null value. GetRule procedure (line 27-37) is used to assign rule for nodes whose parent node is assigned null rule. In GetRule procedure rule for current node is generated by conjunction of nodes till the current node.

Now, each node is checked to see whether there are any nodes whose rule is set as null. The procedure is same as the procedure of RL algorithm except for one thing. Here length criteria are not considered. If rule is not set for a node even after reaching the leaf node then, conjunction of the entire nodes in the branch is performed. After generating the rules the final step is to perform entity resolution. Entity resolution step is same as that of simple rule based ER.

IV. EXPERIMENTAL EVALUATION

In this section, experiments are carried out to compare the two methods. Experiments are conducted on dataset containing medical details of different people. Training set is derived from the dataset according to the parameter given by the user. Algorithms are implemented using java programming and the experiments are conducted on core i5 PC running on windows 7. Experimental results show that the proposed method is more accurate and efficient than the old method.

Using the real dataset, we evaluate (1) the accuracy of both methods (2) the rule creation time of both methods and (3) the impact of training data size on accuracy and rule creation time of both methods.

In the first experiment the accuracy of both methods in two different training percentages is conducted and is reported in Fig. 2. We have the following observations. (1) New method is accurate than the old method. (2) When the training percentage is 35% then accuracy of old method is 68 and the accuracy of new method is 97. (3) Accuracy of old method has increased to 88 and the accuracy of new method have increased to 99 when the training percentage is increased to 50%.

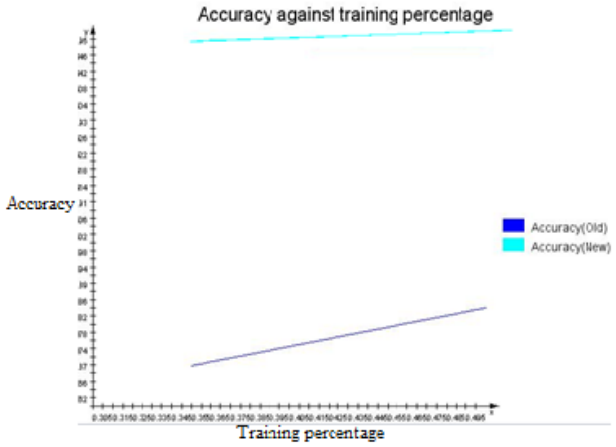

Fig.2.Accuracy against training percentage

In the second experiment effect of two different training data set percentages on rule creation time is examined. The result is reported in Fig. 3. It shows that rule creation time in old method is less when the training percentage is less and it increases when the training percentage increases. In the rule creation time of the new method a different observation is seen. In new method, rule creation time is slightly large when the training percentage is less compared to rule creation time when the training percentage is large. This can be due to the random selection of the training data sets.

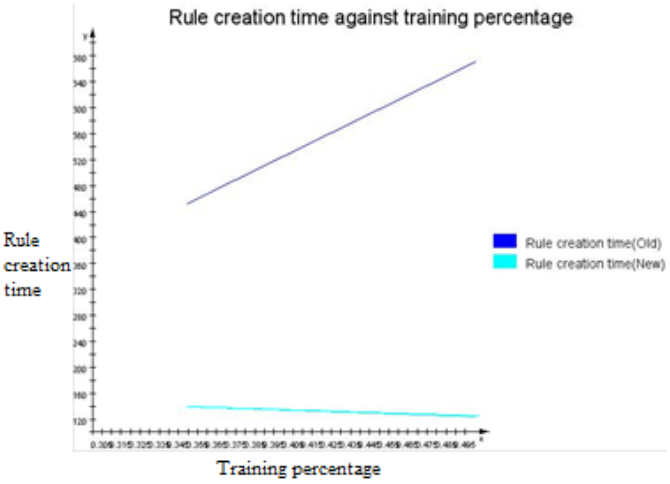

Fig.3.Rule creation time against training percentage

Table I shows the comparison of two methods. Dataset under consideration consists of 768 entries. First the training percentage is set as 35%. Time 1 and Accuracy 1 is the time and accuracy obtained using the existing method. Time 2 and Accuracy 2 is the time and accuracy obtained using the proposed method. Time of the proposed method is less than the time taken by the efficient method. Accuracy of the proposed method is higher than the accuracy of the existing method.

TABLE I
Comparison

datasetsize	Time1	accuracy1	Time2	accuracy2	Training
768	397	0.865885416666667	108	0.958333333333333	0.35
768	564	0.927083333333333	81	0.951822916666667	0.5

V. CONCLUSION

In rule based ER, rules are used to perform ER. In the simple rule based ER method rules are generated without any criteria. This method produces large number of rules during rule generation process and is not efficient. So a new method is proposed that uses distinct tree construction for rule generation. Experimental results show that the proposed method is more efficient and accurate than the old one.

REFERENCES

[1] Lingli Li, Jianzhong Li, and Hong Gao, “Rule-Based Method for Entity Resolution,” IEEE Transactions On Knowledge And Data Engineering, vol. 27, no. 1, january 2015.

[2] I.P. Fellegi and A.B. Sunter, “A Theory for Record Linkage,” J. Am. Statistical Assoc., vol. 64, no. 328, pp. 1183-1210, Dec. 1969.

[3] M. Cochinwala, V. Kurien, G. Lalk, and D. Shasha, “Efficient Data Reconciliation,” Information Sciences, vol. 137, nos. 1-4, pp. 1-15, Sept. 2001.

[4] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, Classification and Regression Trees. CRC Press, July 1984.
[5] S. Sarawagi and A. Bhamidipaty, “Interactive Deduplication Using Active Learning,” Proc. Eighth ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (KDD ’02), pp. 269-278, 2002.