Decision Tree Based Rules for Entity Identification

Shirin Salim
Department of Computer Science and Engineering
Mar Athanasius College of Engineering
Kothamangalam, Kerala,India
shirinsalim93@gmail.com

Linda Sara Mathew
Assistant Professor, Dept. of CSE
Mar Athanasius College of Engineering
Kothamangalam, Kerala,India
lindasaramathew@gmail.com

***Abstract*— Entity resolution (ER) or Entity identification is the process of identifying records referring to the same real world entity. Entity Identification is one of the most important problems in data cleaning and arises in many applications such as information integration and information retrieval. One of the challenges is entity resolution, when integrating data from different sources. As the volume of data on the web or in databases increases, data integration is becoming more expensive and challenging than ever before. For example, different persons may have identical name or other characteristics. So it is necessary to identify such complex records referring to same real world entity. Traditional entity identification approaches obtain a result using similarity comparison among records, assuming that records referring to the same entity are more similar to each other. However, such property may not hold so traditional ER approaches can't identify records correctly in some cases. The proposed framework develops a class of ER rules which are used for entity identification capable to identify the complex matching conditions between records and entities. By incorporating decision tree concept into the rule generation algorithm the proposed framework outperforms the traditional method. In the proposed method, by applying rules to each record, it is possible to identify which entity the record refers to.**

***Keywords*— Entity resolution, rules, length threshold.**

I. INTRODUCTION

Now a days most of the doctors using web search technology for diagnosing complex symptoms in patients. In these cases, they search using patient symptoms without knowing the right diagnosis and then select the most relevant diagnosis from the top results. One of the important problem for analyzing available data is that information is dispersed over many data sources. It is often possible to make use of all the available data most effectively if it is acquired and integrated into a central repository. However integration of the acquired data into a consistent and coherent form is a more challenging problem. In the medical diagnosis example, the doctors had to manually find the most relevant match for each query, but completely manual search is impossible in all but the smallest databases.

Entity Resolution(ER) or Entity Identification (EI) is an important component of data integration that comes up frequently. In many databases, records refer to real world entities, and as such databases grow, there can many different records that refer to the same entity. For example, a social network database can have different records with names `J. Doe', `Jonathan Doe' and `Jon Doe' that refer to the same person or different persons may have same name. In the absence of keys such as social security numbers, these type of issues lead to several different problems, such as redundant records, data inconsistencies, incorrectness of computed statistics, and many others. From a knowledge discovery perspective, mining data that has unresolved duplicates is very likely to yield patterns that are both inaccurate and incomplete. This issue also comes up when integrating data from different heterogeneous sources without shared keys and sometimes different schemas as well.

Entity resolution is a difficult problem and cannot be solved using exact match operators like Cosine similarity, Jaccard similarity etc on tuple attributes. These similarity measures sometimes give high rate of false positives and false negatives. In the proposed work, for the identification of each record efficient rules of attributes are generated from the training dataset which contains different records along with their entity information. These generated rules can be applied to the testing set for the identification and grouping of similar entities. To improve the efficiency of the proposed work, the rules can be generated from decision tree where the attributes are arranged in a particular fashion. So that this method improves the accuracy and saves rule creation time.

Entity is an object which has a physical or logical existence. Relational database is a collection of these entities. Each tuple in relation represent an entity, while each entity may contain one or more tuples. Entity resolution identifies object referring to the same entity. It has application in government, judiciary, public health, shopping etc., There are various task in entity resolution like deduplication, record linkage and reference matching . Deduplication is grouping of records that related to same entity. Record linkage is the linking of records in different datasets. In reference matching incorrect records are match to correct ones in the table. The criterion used in entity resolution is match function. Whether two records are refer to same entity is checked using match function of their one or more attribute values. If the match score is within the threshold, then we said these records are match. Otherwise it does not. The match functions used are exact match, distance, cosine, TF/IDF etc. To reduce the number of pair wise comparison blocking methods [3] are used. The records are divided into blocks based on the blocking key. But it does not sure that all related records are in the same block. So rule based method is used [1]. Here, rules are generated based on attribute-value pairs and apply these rules to different records to find which entity the record refers to.

Traditional Entity Resolution(ER) approaches obtain a result based on similarity comparison among records, assuming that records referring to the same entity are more similar to each other. However, such property may not hold so traditional ER approaches cannot identify records correctly in some cases. Use the following example to illustrate one of the cases. Table 1 shows seven authors with name "wei wang" identified by oij. By accessing to the authors home pages containing their publications, we manually divide the seven authors into three clusters. The records with IDs o11, o12, and o13 refer to the person in UNC, denoted as e1, the records with IDs o21 and o22 refer to the person in UNSW, denoted as e2, and the records with IDs o31 and o32 refer to the person in Fudan University, denoted as e3. The task of entity resolution is to identify e1, e2 and e3 using the information in Table 1. Since all these records have identical name but different set of coauthors in different papers, the similarity between any two records X and Y, denoted by Sim(X,Y), is determined by the similarity of coauthors. To measure the similarity between sets, Jaccard similarity is often used, and thus the similarity between any two records, X and Y , is defined as:

$$Sim(x,y) = \frac{\left| coauthors(x) \cap coauthors(y) \right|}{\left| coauthors(x) \cup coauthors(y) \right|} \quad (1)$$

Thus, we have the following facts:
Sim(o11,o12)<Sim(o11,o31) since Sim(o11,o12)=0 and Sim(o11,o31)= 0.5
Sim(o12,o13) < Sim(o12,o21) since Sim(o12,o13)=0.2 and Sim(o12,o21)= 0.25

The results show that the similarity between o11 and o12 is smaller than the similarity between o11 and o31 even though o11 and o12 refer to the same entity while o11 and o31 refer to different entities. It is obvious that we are unable to get the correct ER result of the example by applying similarity comparison between records. Similar to Jaccard, other similarity functions, such as cosine similarity also have the same problem.

As similarity comparisons cannot be applied in this case, the results show that the existence of some attribute value pairs are useful to identify records as well as some non existence of attributes are also helpful. Take Table1 as an example, the attribute-value pair (coauthors,"lin") occurs only in the records referring to e2. Thus, the existence of (coauthors, "lin") can be used to identify records referring to e2. Similarly, the existence of (coauthors, "kum") and (coauthors, "shi") can be used to identify records referring to e1 and e3 respectively. The nonexistence of some attribute value pairs are also useful to identify records. Taking Table 1 as an example again, the coauthors of the record o11 includes only "zhang". Since "zhang" occurs in both o11 and o31, the existence of (coauthors,"zhang") can distinguish the records referring to e1 or e3 from the other records, but cannot distinguish the records referring to e1 and e3. However, the nonexistence of (coauthors, "shi") can be used to rule out the possibility of o11 referring to e3 since the existence of (coauthors, "shi") can identify all the records referring to e3. Thus, the existence of (coauthors, "zhang") and the nonexistence of (coauthors, "shi") can be used together to identify the records referring to e1.So the rules look like:
R1: (name ="wei wang") Λ ("zhang" Ɛ coa) Λ --("shi" Ɛ coa)) => e1
R2: (name ="wei wang") Λ ("kum" Ɛ coa)) =>e1,
R3: (name ="wei wang") Λ ("lin" Ɛ coa)) =>e2,
R4: (name ="wei wang") Λ ("shi" Ɛ coa)) => e3,

TABLE 1: Paper – Author Records

Id	Name	Co authors	Title
O11	Wei Wang	Zhang	Inferring……
O12	Wei Wang	Dunkan, Kum, Pei	Social……
O13	Wei Wang	Cheng, Lie, Kum	Measuring….
O21	Wei Wang	Lin, Pei	Threshold….
O23	Wei Wang	Lin,Hua,Pei	Ranking…….
O31	Wei Wang	She, Zhang	Picture book….
O32	Wei Wang	Pei, Shi, Xu	Utility……

Syntax for rule – The rule has both LHS and RHS. LHS represents the entity and RHS denotes the conjunction of clauses that can be used to identify the entity. Clause is the combination of attribute and it's value. Rule is in the form shown below.
e1⟹ c1∧ c2∧……. ∧ci
Coverage-It used to find the validity of the rule. Coverage of a rule is the objects that can be identified by satisfying the RHS of the rule.
Validity-Rule is invalid if it has coverage in another entity; otherwise it is a valid rule.
Length requirement- It is used to decrease the number of clauses in the rule.
Distinct (Decision) Tree- Tree is generated using various attribute and its values for rule generation. It is formed by choosing the attribute, which have least number of distinct value as the parent node. This method is chosen to reduce the complexity of rule generation.

II. PROPOSED METHOD

Existing system generates rules and applies to records for entity identification. First the entity set is found, and then we obtain a training set from entity set for rule generation. Rule sets are generated entity wise. Each attribute-value is considered as a single rule. Coverage of a rule is the objects that can be identified by satisfying the clauses of the rule. Rule is invalid if it has coverage in another entity; otherwise it is a valid rule. The validity of above rules are checked. If valid, then stored in R and if invalid, then stored in the arrays LX and LY. LY consists of all the generated invalid rules. LX consists of invalid rule with length 1. Length is provided by the user. Length is used to decrease the number of attributes in the rule. After the first round of rule generation, check the length threshold. If within the threshold, conjunction of LX & LY is performed and checks its validity. If valid add to R, else add to LY. Then again check the length threshold of newly

generated rules in LY .We continue conjunction of LX & newly generated rules of LY until the length threshold is met. Now we test whether, it is possible to identify all the objects in training set using the rules in R. If any object are left out, then create rule for each object by conjunction of there entire attribute-value. Reduce the length of rule by eliminating each attribute-value, until it becomes invalid. There can be possibility to find one object from one or more rules .So we reduce the number of rules by greedy algorithm. In this select those rules that can be used to find more than a single object.

The main drawback of rule creation proposed in R-ER[1] is that, when the database size increases it leads to more rules and become complex. It simply generates rules by just considering each attribute value pair combinations. In order to overcome this limitation our proposed method generates rule by using the concept of decision tree. Instead of simply taking each attribute value combinations, it identifies the most distinctive and useful attribute value pair. So that it leads to better rules with minimum time.

III. RULE DISCOVERY

A. Entity Set Creation

In this entity set is generated from raw dataset. We manually create the entity set using one or more attribute in the raw dataset.

B. Training Set Creation

Training set is generated by random sampling of records from the entity set according to a parameter. Parameter can be any number of records from each entity set. For example, 20% of records can be taken from each entity and the number of entity is 10. Let T be the training set and E the entity. Therefore Ti = max{0.2|Ei|,1}.

PR(Positive Rule) is an ER-rule which only includes positive clauses. NR(Negative Rule) is an ER-rule which includes at least one negative clause.

Length Requirement: Given a threshold l, each rule r in R satisfies that |r|<=length treshold. To determine whether record o matches the LHS of ERrule r, we should check whether o satisfies each clause in LHS(r). Thus to guarantee the efficiency of rule-based ER (R-ER) and avoid overfitting, the length of each rule (the number of clauses) should be no more than a threshold.

Algorithm : Rule Discovery
Input : length threshod l, training set S={ s1,s2,….s3}
Output: ER Rule Set R
1: S= S1 U S2 U. . . . Sm
2: R= Gen_PR (l,S)
3: If Coverage(R)≠S then
4: S'=S ─ Coverage(R)
5: for each row in S' do
6: If r0 is valid then
7: R.insert(Min_Rule(r0))
8: else
9: R.insert (Gen_NR(o))
10: end if
11: end for
13: end if

Algorithm : Gen_PR (l,S)
Input: length threshold l, input set S
Output: positive rules
1: for each sj in S do
2: for each attribute value pair t in sj do
3: if Cov (t) = Sj do
4: Rj.insert(r(t))
5: else
6: Lx.insert(r(t))
7: Ly.insert(r(t))
8: end if
9: end for
10: for each ri from Ly do
11: if |ri|>l then
12: break
13: end if
14: for each rk from Lx
15: if Cov(ri Λ rk) ≠Φ then
16: if Cov(ri Λ rk) =Sj then
17: Rj.insert(ri Λ rk)
18: else
19: Ly.insert(ri Λ rk)
20: end if
21: end if
22: end for
23: end for
24: end for
25: R= R1UR2U…Rm
26: Return R

PR Requirement: each rule r in R is a PR. The reason why we give priority to PRs is that, positive literals lead to bounded spaces while negative literals lead to unbounded spaces. Therefore the discovered PRs are more possible to identify other data sets effectively than the discovered NRs. However, both the length requirement and the PR requirement can decrease the expression power of ER-rules, so that there might be some records in S that cannot be identified by any valid PR with length no more than l. To guarantee the completeness, the requirements should be relaxed. Specifically, for each record o in S that cannot be identified by any valid PR with length no more than l, a valid PR with the smallest length that identifies o should be discovered; if no valid PR can identify o, a valid NR with the smallest length that identifies o should be discovered.

Step 1. Generate atomic PRs (lines 2-9). To find all the preliminary and valid atomic PRs of ej, for each attribute-value pair t that appears in the records in Sj, we should check whether the corresponding rule is preliminary or valid. If it is valid then it is added to Rj (lines 3-4); otherwise it is added to both Lx and Ly for further conjunction (lines 5-7).

Step 2. Generate PRs with length >1 (lines 10-23). In this step, we conjunct each preliminary rule ri in Lx(line 8) with each

preliminary rule rk in Ly (line 9) to generate a new PR, ri ^ rk. This new rule is added to Rj if it is valid and its coverage is not empty (lines 10-12); or added to Ly for further conjunctions if it is preliminary (lines 13-14); otherwise it is useless and can be ignored.

Algorithm : Gen_NR
Input: Record O, that is not satisfied by positive rules R
Output: Negative Rules
1: T←0
2: for each Oi do
3: for each tj in Oi do
4: if O satisfies –tj then
5: T.insert(-tj)
6: end if
7: end for
8: end for

Algorithm: Min Rule
Input: ER rule R
Output: Minimum Rules
1: for each clause ti in R do
2: let r' be the sub rule of r that excludes ti
3: if Cov(r) = Cov(r') then
4: r ← r'
5: end if
6: end for

Given an ER-rule r, Min_Rule checks each clause Ti in r to identify whether Ti can be removed . If so, Ti is removed from r. Given a valid ER-rule r, Min_Rule outputs a minimal sub-rule r0 of r.

IV. EFFICIENT RULE DISCOVERY

By using the concept of decision tree , it will produce a better result.
Algorithm 1 : Rule Discovery
Framework Input : length threshold l,
Training set T ={T1 ,T2 …..Tn }
Output: Rule set R
1: T ← {T1 ,T2 ……..Tn}
2: TTree← DistTreeGen(T)
3: Generate Rules with threshold length
4: if records not satisfies by the rules R then
5: generate rules without length threshold
6: end

The attributes are arranged in a particular fashion and generate decision tree and select the most distinctive attribute as initial attribute value. That is, the attribute with least distinct values are taken as initial attribute and then rules are generated. If these rules cannot satisfies all the records, then next least distinct valued attribute is selected and continue rule generation until the generated rules can satisfies all the records. Traverse the decision tree in order to get the rules. If the records cannot satisfied by these rules then relax the length threshold.

V. EXPERIMENTS

Experiments are conducted to determine the accuracy, false negative and time required to complete the program of the proposed algorithm. Dataset contain medical details of different persons. Training set is derived from the dataset according to the parameter given by the user. The algorithms are implemented using java programming. The experiments are performed on a corei3PC, running in windows 7. Proposed method is compared against R-ER [1].

To ensure fair comparison with R-ER[1], the accuracy measure F- measure is used. It is the harmonic mean of recall and precision. Precision representing number of tuples correctly identified to the number of tuples return by method. Recall represents number of tuples correctly identified to the number of relevant tuples. The comparison results are shown in table 2. From the table it is cleared that even the training size increases the proposed method gives more accuracy than old method. Figure 1 shows the graph plotted against these values.

Also from the results for rule creation time and false negatives in table 3 and table 4, we reached the conclusion that, our proposed method outperforms the old method. The rule creation time is less for new method compared to old method. Figure 3 and 4 represents the rule creation time and false negative against training percentage respectively.
Accuracy = (TP+TN)/(P+N) where TP represents true positives , TN represents true negatives, P=TP+FN and N= FP+TN.

TABLE 2. False Negative

Training Percentage	R_ER(Old Method)	Decision Tree Based Method
10	0.58	0.2
15	0.48	0.25
20	0.35	0.18
25	0.3	0.15
30	0.28	0.13
35	0.2	0.12

TABLE 3. F-Measure

Training Percentage	R_ER(Old Method)	Decision Tree Based Method
10	0.31	0.4
15	0.38	0.48
20	0.48	0.52
25	0.56	0.6
30	0.62	0.62
35	0.68	0.72

TABLE 4. Rule Creation Time

Training Percentage	R_ER(Old Method)	Decision Tree Based Method
10	190	100
15	250	105
20	400	200
25	565	210
30	680	220
35	800	350

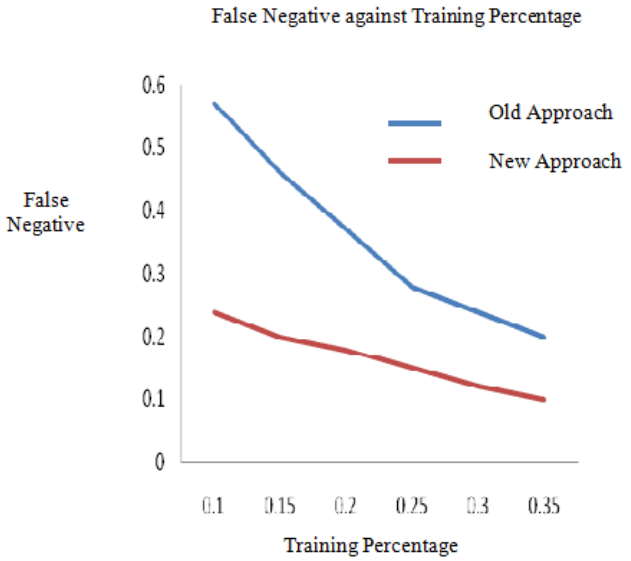

Fig 1. False Negative against Training Percentage

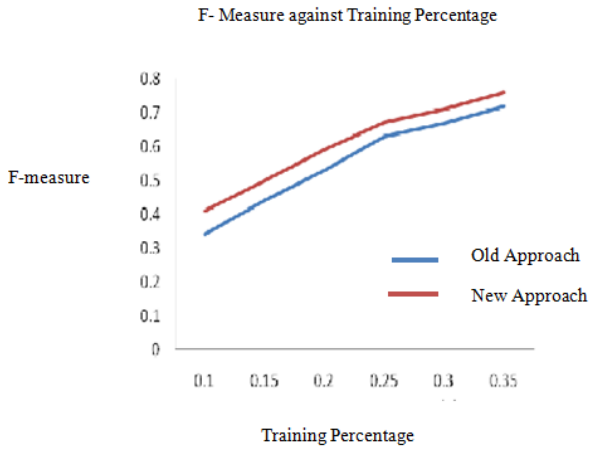

Fig 2. F-measure against Training Percentage

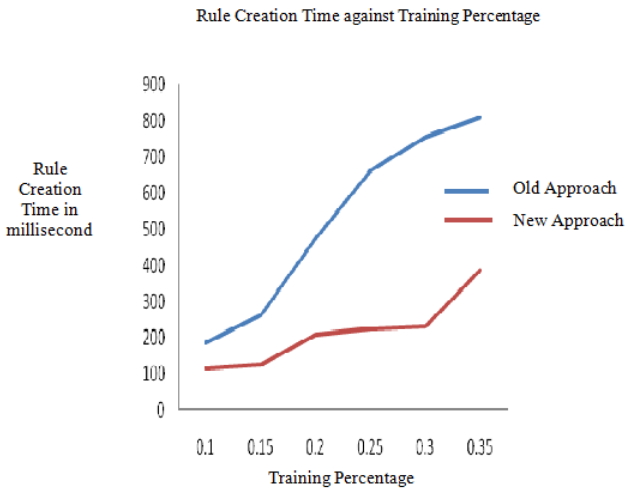

Fig 3. Rule creation time against training percentage

VI. CONCLUSION

Entity resolution identifies object referring to the same entity. Entity resolution is performed by generating rules from training set and applies them on records. Traditional ER considered each attribute value as the rule in a random fashion and performs conjunction with other rules according to length threshold. This method is very complex and tedious. Our proposed method generated rules from a decision tree. Decision tree is formed by arranging attribute and its values of records in the training set in a particular fashion. These generated rules are applied to the dataset for entity identification. The experimental results show that the proposed method is more accurate.

REFERENCES

[1] Lingli L, 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] Chaudhuri, V. Ganti, and R. Motwani, "Robust identification of fuzzy duplicates," in Proc. 21st Int. Conf. Data Eng., 2005, pp. 865–876.

[3] Sheila Tejada. , Minton,"Learning Object Identification Rules For Information Integration" Information Systems vol. 26, no. 8, pp. 607-633, 2001.

[4] M Ganesh and Travis Richardson"Mining Entity-Identification Rules For Database Integration", KDD-96 Proceedings,1996.

[5] L. Shu, B. Long, and W. Meng "A latent topic model for complete entity resolution," in Proc. 25th Int. Conf. Data Eng., 2009, pp. 880- 891.

[6] S.E. Whang and H. Garcia-Molina, "Entity resolution with evolving rules," Proc. VLDB Endowment, vol. 3, no. 1, pp. 1326–1337, 2010.

[7] Alvaro E. Monge and Charles P. Elkan ,"The field matching problem: Algorithms and applications" KDD-96 Proceedings,1996.