R:Incremental entitiy resolution on rules and data

February 12, 2014

1 Introduction

This paper invesitgates how to enhance the performance of Entity Resolution on effience, but not on accuracy or recaLl. Specifically, it first address the problem of keeping the ER result up-to-date when the ER logic or data "evolve" frequently. It says the naive approach for ER is to re-run ER from scractch, which is not tolerable for large datasets. The author's main idea is some like to use Greedy Algorithm and "Divide and Conquer" strategy for saving redundant work of evolution by exploiting previous *"materialized"* ER results.

The paper focuses on two types of evolution: (1)Rule Evolution (2)Data Evolution. For (1), the solution given is to *"materialize"* the previous results. When rule B_1 is improved yielding rule B_2, the preivous results based on B_1 could be used directly to help computing the new ER result obeying rule B_2. To illustrate, consider this case: let $B_1 = p_{name}$, say that two records match if predicate p_{name} evaluates to true, and $B_2 = p_{name} \wedge p_{zip}$, which is stricter than B_1. When the algorithm calls on B_1 to compare records in set S, say, $\{\{r_1\}, \{r_2\}, \{r_3\}, \{r_4\}\}$, we get the result $\{\{r_1, r_2, r_3\}, \{r_4\}\}$. Because the evolved rule B_2 is stricter than B_1, the second ER based on the new rule can be computed directly from the previous result rather than the initial set S. Specifically, we can just run $B = p_{zip}$ on $\{\{r_1, r_2, r_3\}, \{r_4\}\}$ to reach the final result. From this case, it's obvious to see that materializing the previous result can help to simplify the new complicated rule and bring in cost savings from the intermediate results. But if the evolved rule is not quite stricter than the old rule, the process becomes more challenging. When is materialization feasible? Which scenarios can it pay off? Under what conditions and for what strategies are the savings over the naive approach significant? Above problems are the important part of this paper. Compared to (1), the problem (2) is rather trivial. We can just resolve the new incoming data and old data sparately, and utilize a little skill on the combination step. What's more, generally, the new data is always extremely small than the old data, so its overhead is not able to influence the system too much.

2 Model and properties

The paper considers three ER models: match-based clustering, distance-based clustering and pairs ER. The author mainly focuses on the match-based clustering model, using a Boolean comparison rule for resolving records. Because he shows that there are not much changes for the properties among these three models.

2.1 Match-based clustering model

Define a Boolean comparison rule B as a function that takes two records and returns `TRUE` or `FALSE`. Assume that B is commutative, i.e., $\forall r_i, r_j, B(r_i, r_j) = B(r_j, r_i)$. An ER algorithm receives as inputs a partition P_i of S and a Boolean comparison rule B and returns another partition P_o of S. A partition $P = \{c_1, ..., c_m\}$ such that $c_1 \cup ... \cup c_m = S$ and $\forall c_i, c_j \in P$ where $i \neq j, c_i \cap c_j = \emptyset$. For example, when $P_i = \{\{r_1, r_2, r_3\}, \{r_4\}\}$, the output using the the rule $B = p_{name} \wedge p_{zip}$ is the partition $P_o = \{\{r_1, r_2, r_3\}, \{r_4\}\}$. However, we should notice three places in this model. First, not all the ER algorithm simply cluster records based on B. Second, input clusters are allowed to be un-merged as long as the final ER result is stll a partition of the records in S. Un-merging means incorrect clustering occurs. Third, the ER algorithm is *nondeterministic*. A hierarchical clustering algorithm based on Bollean rules may produce different partitions depending on which records are compared first. Formally, denote an ER result $E(P_i, B)$ produced by input partition p_i and rule B. Denote all the possible paritions that can be produced by the ER algorithm E as $\bar{E}(P_i, B)$. For example, given $P_i = \{\{r_1\}, \{r_2\}, \{r_3\}, \{r_4\}\}$, $\bar{E}(P_i, B)$ could be $\{\{\{r_1, r_2, r_3\}, \{r_4\}\}, \{\{r_1, r_2\}, \{r_3, r_4\}\}\}$ while $E(P_i, B) = \{\{r_1, r_2, r_3\}, \{r_4\}\}$. Another important concept is strictness. A Boolean comparison rule B_1 is *stricter than* rule B_2(denoted as $B_1 \leq B_2$) if $\forall r_i, r_j, B_1(r_i, r_j) =$`TRUE` implies $B_2(r_i, r_j) =$`TRUE`.

How does the rule generate? Though this part is significant, the author's answer is *unconvincing*. He says, Machine Learning, probabilistic modeling, and graph-based approaches can be used to develop and refine rules. If ML is used to determine what features and thresholds to use for deciding if records match, there will be no evolution on the rule. Unless new data added into training sets, the rule learned by ML will not change. Because the rule is learned from a fixed training data set. Thus, problem(1) is redundant. What's more, the cost of learning a rule plus computing a result is definitely larger than learning a result directly. The only advantage of the author's method is that it can proceed newly small added small data in real time.

2.2 Properties

This part is the most important of the paper. It introduces two new properties for ER algorithms - *rule monotonic* and *context free* and two existing properties - *general incremental* and *order independent*.

2.2.1 Rule monotonic

First, define the notion of refinement between partitions. A partition P_1 of a set S *refines* another partition P_2 of S(denoted as $P_1 \leq P_2$) if $\forall c_1 \in P_1, \exists c_2 \in P_2$ s.t. $c_1 \subseteq c_2$. Now define the rule monotonic property, which says that the stricter the comparison rule, the more refined the ER result. For example, suppose that $P = \{\{r_1\}, \{r_2\}, \{r_3\}, \{r_4\}\}, B_1 \leq B_2$, and $E(P_i, B_1) = \{\{r_1, r_2, r_3\}, \{r_4\}\}$. If the ER algorithm is $\mathcal{RM}$, $E(P_i, B_2)$ can only return $\{\{r_1, r_2, r_3\}, \{r_4\}\}$ or $\{\{r_1, r_2, r_3, r_4\}\}$.

2.2.2 Context free

This property says that a subset of P_i can be processed "in isolation" from the rest of the cluster. For clarification, just imagine that in a set of animals, the cluster of birds will no longer have an intersaction with the cluster of dogs. With such knowledge, we can separately resolve the mutual independent subsets. The key of this property is that it requires the experts have the related pri-knowledge. Formally, with knowledge that no clusters in $P = \{\{r_1\}, \{r_2\}\}$ will merge with any of the clusters in $P_i - P = \{\{r_3\}, \{r_4\}\}$ and for any $P_o \in \bar{E}(P_i, B), P_o \leq \{\{r_1, r_2\}, \{r_3, r_4\}\}$, we say the ER algorithm is $\mathcal{CF}$ for such P_i, P and B. In this case, the algorithm can resolve P independently from $P_i - P$, and there exists an ER result of P_i that is the same as the union of the ER resuls of P and $P_i - P$.

2.2.3 General incremental(more natural)

This property is similar to the $\mathcal{CF}$, but looser than that. The main difference is that the $\mathcal{CF}$ requires the subsets to be independent, but the $\mathcal{GI}$ does not.The GI says that we can resolve P fisrt. Then we can add the result of $E(P, B)$ to the remaining clusters and resolve all the clusters together. The its idea to save costs is just like Merge Sort.

2.2.4 Order independent

This properties says that an ER algorithm is $\mathcal{OI}$ if the ER result is same regardless of the order of the records processed. In other words, the ER algorithm is *deterministic*. $\bar{E}(P_i, B)$ contains exactly one partition of S.

3 Rule evolution

3.1 Materialization

Though when and how to materialize earilier results is a big problem, the author's answer doesn't make sense. He assumes that all rules are in conjunctive normal form, like $B = p_1 \wedge p_2 \wedge (p_3 \vee p_4)$. Certainly, if we know how the rules generate and evolve, the most useful conjuncts become easier to find for materializing. In the last section, he said ML could be used to determine the rules, while in this section he used the most naive method, materializing all conjuncts of B, instead of technical approachs. Though he could amortize the common costs in a concurrent fashion, it is still not persuasive.

3.2 Rule evolution technique

First, the author introduces a new definition *meet*. The *meet* of P_1 and P_2(denoted as $P_1 \wedge P_2$) returns a new partition of S whose members are the non-empty intersections of the clusters of P_1 with those of P_2.

3.2.1 ER algorithm satisfying $\mathcal{RM}$ and $\mathcal{CF}$

Suppose that $B_1 = P_1 \wedge P_2 \wedge P_3$ and $B_2 = P_1 \wedge P_2 \wedge P_4$. When B_1 evolves into B_2, we first materialize the common conjuncts of B_1 and $B_2 -$ "$p_1 \wedge p_2$". The reason to materilize the common conjuncts is to utilize the property $\mathcal{RM}$. Because B_2 is stricter than any conjuncts of itself and basde on $\mathcal{RM}$, $E(P_i, B_2)$ could be directly computed from the previous result $E(P_i,$ "$p_1 \wedge p_2$"). Then exploiting the property $\mathcal{CF}$, we can resolve each cluster of the result $E(P_i,$ "$p_1 \wedge p_2$") independently using the rule B_2(the reason not to use "p_4" is unknown. Maybe depend on the algorithm). According to $\mathcal{RM}$ and $\mathcal{CF}$, while the complexity of the algorithm is not improved, lots of redundant work are saved by running ER on small clusters of the previous materialized results.

3.2.2 ER algorithm satisfying $\mathcal{RM}$ only

Without $\mathcal{CF}$, the subsets of P_i can not be processed independently. It doesn't mean that We must treat the materialized previous result as an entire set to handle. It implies when processing one subset we must put its relatedly dependent subset into consideration. Suppose we have an intermediate subsets $\{r_1, r_3, r_5\}, \{r_2\}$ and $\{r_4\}$. $B_2(r_1, r_3)$ =`FLASE`, $B_2(r_3, r_5)$ =`FLASE` and $B_2(r_1, r_5)$ =`TRUE`. But the algorithm rules that only the records within a sliding window of size 3 on the sorted list of records can be compared(r_1, r_5

only match during the transitive closure). Thus, when computing $\{r_1, r_3, r_5\}$, we must take $\{r_2\}$ and $\{r_4\}$ into account. With $\{r_2\}$ and $\{r_4\}$, the distance between $\{r_1\}$ and $\{r_5\}$ becomes 4 but not 2. Thus, r_1, r_5 will not be clustered in the final result even though $B_2(r_1, r_5) =$`TRUE`.

3.2.3 ER algorithm satisfying $\mathcal{GI}$

Though no longer satisfying $\mathcal{RM}$ and $\mathcal{CF}$, the algorithm is still able to make use of the materializations of the common conjuncts. But this time, we must treat the materialized previous result as an entire set to handle. It is still efficient for the same reason of Merge Sort.

3.3 Materialization strategies

This section also makes nonsense. If a group of conjuncts appears together frequently in most rules, they should be materialized; If the rules are looser than other rules, they should be materialized.

4 Data Evolution

The method to handle the newly incoming data is similar to Rule Evolution. We can just treat the new dat as a subset of P_i. Then the algorithm and startegies in the last section can all be applied.

5 Variations

This section shows that the properties and evolution algorithm mentioned in previous sections can carry over to Distance-based evolution and Pairs ER evolution. The only tricky place is in the Distance-based model part. Denote D as a commutative distance function that returns a non-negative distance between two records and d as a distance between two records on a specific attribute. Suppose $D_1 = d_1 + d_2$ and $D_2 = d_1 + d_3$, how to express the "common conjuncts" between D_1 and D_2? Restrict d_2 and d_3 distances to be at most f. When evolve from D_1 into D_2, define a third distance comparison rule $D_3(r, s) = [max\{(D_1(r, s).min - f), 0\}, D_1(r, s).max + f]$. As a result, D_3 acts as the "common conjuncts".

6 ER algorithms and their properties

This section summarizes some important algorithms and their properties, including the sorted neighborhood algorithm(SN), Hierarchical clustering based on a Boolean comparison(HC_{BR}), Monge-Elkan(ME), and the complete-link hierarchical clustering algorithm(HC_{DC}).

7 Experimental evaluation

The author evaluates the performance of his algorithm on three metrics: CPU, IO, and storage costs. As his techniques do not change the ER result, he does not consider accuracy. And also there is not a recall problem. Because the result is always a partition of S with fixed size. Ml is not used in the experiment as the author said in previous sections.

8 Papers worth reading

Blocking: [25]
Matched based clustering: [3,16]
Distance-based clustering: [4,21]
Pairs ER: [2,26,31] (active learning)
Orthers: [7,10]