A Query-driven and Incremental Process for Entity Resolution

Priscilla Kelly M. Viera[1, 2], Ana Carolina Salgado[1], Bernadette Farias Lóscio[1]

[1] Federal University of Pernambuco, Center for Informatics, Recife, Pernambuco, Brazil
{pkmv, acs, bfl}@cin.ufpe.br

[2] Federal Rural University of Pernambuco, Recife, Pernambuco, Brazil

1 Introduction

Companies and governmental organizations around the world publish a huge volume of data, which can be stored in multiple data sources. In order to access and analyze these data, strategies for data integration are needed. The aim of data integration is to combine heterogeneous and autonomous data sources for providing a single view to the user [1]. An important component of the data integration process is the Entity Resolution (ER) task [2]. The ER goal is to identify tuples referring to the same real-word entity (in this work, tuple is synonymous of instance and record). This problem is known by a variety of names: Record Linkage, Entity Resolution, Object Reference, Reference Linkage, Duplicate Detection or Deduplication. In this paper, we adopt the term Entity Resolution (ER) [2].

Often, companies and organizations have to deal with dynamic data sources with a large volume of data. In this context, the ER process can be very challenging because most current available ER techniques process all the entities at one time [3]. This occurs because most of these techniques are based on batch algorithms, which resolve all tuples instead of resolving those related to a single query [4, 5, 6]. Then, arises the need of new techniques to support real-time ER for dynamic and large databases.

For example, suppose a set of data sources of bibliographic data and a query to retrieve all papers from a given author (e.g. "Getoor"). To answer this query, it is not necessary to look for other author's papers and to perform the ER considering the whole set of papers. In this case, it would be better to focus on the tuples describing just papers from the author specified in the query.

In this paper, we propose a QUery-Driven and Incremental process for Entity Resolution (QuID). The QuID process considers query results on multiple data sources. It is an incremental process, i.e., for each new query result, QuID reuses the previous ER clusters to answer future queries. In our approach, ER is considered as a clustering problem [7], in which each cluster corresponds to tuples of a single real-world entity. During the ER, the results of queries are analyzed, and each tuple of the query result is inserted incrementally in a cluster. Our solution holds an index for the tuples, and performs incremental clustering, resulting in clusters of tuples that refer to the same real-world entity. The rest of the paper is organized as follows. In Section 2 we discuss related work. In Section 3 we formally define the problem and describe the QuID process and in Section 4 we conclude.

2 Related Work

Bhattacharya and Getoor [4] proposed a strategy adjusted for query-time entity resolution by identifying and resolving only those database references that are the most helpful for processing a given query. Altwaijry [5] proposed a query-driven approach to ER, exploiting the specificity and semantics of the given SQL query. Both papers do not propose to reuse previous results of the ER process. The solution proposed by Gruenheid [3] uses an incremental clustering algorithm to perform ER. Each inserted tuple is compared with existing clusters, either putting the tuple into an existing cluster, or creating a new cluster for it, using extra information from the data updates to fix previous cluster problems. This solution does not consider query results during the ER task. Different from the mentioned approaches, the process proposed in this paper is incremental and query-driven. To the best of our knowledge there are no other approaches that combine these two features.

3 Problem Statement

In this section we formally define the problem of query-driven and incremental ER (Section 3.1). We then describe our Query-Driven and Incremental process for Entity Resolution (QuID) (Section 3.2).

3.1 Problem Definition

Given a set of tuples, the ER process is essentially a clustering problem, in which each cluster contains tuples that represent a single real-world entity. If we consider the ER problem in multiple data sources, each tuple can be from a different source.

In this paper, our focus is on incremental clustering algorithms. The goal of the incremental clustering approach is to make the ER process faster than other processes that do not use this strategy. The main goal of using the query results is to reduce the volume of tuples. This strategy will also reduce the number of comparisons made between tuples.

Formally, we denote $S = \{S_1, S_2, ..., S_n\}$ a set of data sources and $Q = \{Q_1, Q_2, ..., Q_m\}$ a set of queries running on S. Each source has a set of entities $S_i.E$, where $E = \{E_1, E_2, ..., E_w\}$. Each entity E_j from $S_i.E$ has a set of tuples $S_i.E_j.T = \{t_1, t_2, ..., t_n\}$, where each t_p is an instance of the entity E_j. A tuple t_p is defined as follows.

Definition 1. Each **tuple** t_p belonging to $S_i.E_j.T$, is represented by a set of pairs of attributes (A_k) and values (v_k), $t_p = \{(S_i.E_j.A_1, v_1), (S_i.E_j.A_2, v_2), ..., (S_i.E_j.A_n, v_n)\}$. Each attribute A_k belongs to an entity (E_j) of a data source (S_i), denoted by $S_i.E_j.A_k$. Each tuple t_p has a pair $(S_i.E_j.A_k, v_k)$, which represents a single identifier of the tuple (*Id*).

A query Q_i may not contain all the attributes necessary (relevant) to define whether two tuples represent the same real-world entity. Thus, the query is submitted to an expansion process for collecting the relevant attributes [8] that were not informed in the initial query. This expansion generates a query Q_i'. The input of the QuID process is the result of the query Q_i', defined as follows.

Definition 2. A **query result**, $Q_i'.R$, is represented by a set of tuples (Definition 1) that belongs to an entity E_j. The attributes that describes the tuples of the result $Q_i'.R$ includes the set of relevant attributes (A^r), $S_i.E_j.A^r$, where $S_i.E_j.A^r \subseteq S_i.E_j.A$.

For each new received query result, the ER process reuses the results of previous ER tasks, i.e., previous generated clusters, to respond the query.

3.2 QuID

In this section, we describe the proposed process (QuID). Fig. 1 shows the flow of information in QuID. The input of the process is a query result ($Q'_i.R'$). The process starts with the Indexing step, which aims to reduce the number of comparisons between pairs of tuples. During this step, two indexes are used: the Similarity Index and the Cluster Index. The first one maintains incrementally the similarity values between each pair of tuples. The second one maintains incrementally a set of clusters of tuples identifiers.

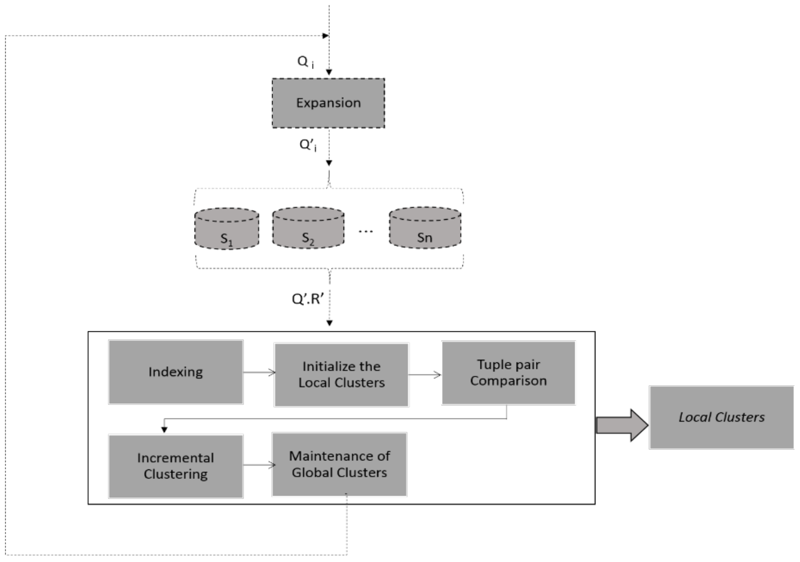

Fig. 1. Proposed process (QuID)

Our approach, uses two types of clusters: global clusters and local clusters. Global Clusters (G_c) are created only once and updated, incrementally, at each query result $Q_i'.R'$. A G_c offers support to the query-driven process reusing previous results in future queries. A global cluster is defined in the following.

Definition 3. A **Global Cluster** (G_c) is defined by a set of triples, $G_c = \{(ClusterId, S_i.E_j,\ S_i.E_j.t_p.Id)\}$, where *ClusterId* is an identifier of the cluster, $S_i.E_j$ is the entity and the data source of the tuple t_p and $S_i.E_j.t_p.Id$ is the tuple identifier.

Local Clusters (L_c) are created for each query result $Q_i'.R'$. The output of the ER process is the L_c containing the duplicated tuples detected in the query result. L_c will use previously classified information from the global cluster G_c. We define local cluster as follows.

Definition 4. A **Local Cluster** (L_c) is defined by a set of pairs, $L_c = \{(S_i.E_j.t_k, ClusterId)\}$, where $S_i.E_j.t_k$ is a tuple and *ClusterId* is the identifier of the cluster which the tuple belongs to.

After the Indexing step, the local cluster (L_c) is initialized from G_c, reusing the results of previous ER tasks. After the initialization of L_c, the tuples not processed previously will be processed during the Tuple Pair Comparison step. In this step, similarity values are recovered from the Similarity Index, or new similarity values between two tuples are calculated.

After the Tuple Pair Comparison phase, the next step is the Incremental Clustering. The input of this task is a similarity graph, where nodes are tuples, and similarity values between tuples are edges. The goal of the Incremental Clustering is to insert into the local cluster (L_c) and global cluster (G_c) the tuples not processed before. Finally, after the Incremental Clustering, the output of QuID is L_c and G_c already updated for reuse in the next ER tasks.

4 Conclusions

In this short paper, we introduced and motivated an incremental and query-driven Entity Resolution process, denoted QuID. We also presented the main components of QuID and some important definitions related to our proposal. In the current state of our work, we implemented the two proposed indexes (cluster index and similarity index). Currently, we are investigating and evaluating the impact of the incremental clustering algorithm [3, 4] in the context of the proposed process. As future work, we will instantiate and evaluate the complete process.

References

1. Lenzerini, M. Ontology-based Data Management. In: international conference on Information and knowledge management (CIKM'11). New York, NY, USA, pp. 5-6, 2011.
2. Christen, P. Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer. 2012.
3. Gruenheid, A.; Dong, X. L.; Srivastava, D. Incremental Record Linkage. In: VLDB'2014. Hangzhou, China. 2014.
4. Bhattacharya, I., Getoor, L. Query-time Entity Resolution. Journal of Artificial Intelligence Reserche. 2007.
5. Altwaijry, H., Kalashnikov, D. D., Mehrotra, S. Query-Driven Aproach to Entity Resolution. VLDB 2013, Italy. 2013.
6. Su, W., Wang, J., Lochovsky, F, H. Record Matching Over Query Results from Multiple Web Databases. IEEE Transactions on Knowledge and Data Engineering. Vol. 22, No. 4. 2010.
7. Berkhin, P. A Survey of Clustering Data Mining Techniques. Grouping Multidimensional Data: Recent Advances in Clustering. Pp 25 – 71. Springer Berlin Heidelberg. 2006.
8. Whang, S. E.; Marmaros, D.; Garcia-Molina, H. Pay-As-You-Go Entity Resolution. In: IEEE Transactions on Knowledge and Data Engineering. Volume 25 Issue 5. 2013.