Efficient and Effective On-line Learning-Based Instance Matching over Heterogeneous Data

Samur Araujo #1, Duc Thanh Tran *2, Arjen de Vries #3

#Delft University of Technology, PO Box 5031, 2600 GA Delft, the Netherlands
1s.f.cardosodearaujo@tudelft.nl
3a.p.devries@tudelft.nl

*Karlsruher Institute of Technology, Germany
2ducthanh.tran@kit.edu

Abstract— **This document gives formatting instructions for authors preparing papers for publication in the Proceedings of an IEEE conference. The authors must follow the instructions given in the document for the papers to be published. You can use this document as both an instruction set and as a template into which you can type your own text.**

I. Problem Definition

In this paper, we tackle the problem of candidate selection queries over multiple heterogeneous datasets where there might be only little or no overlap between schemas. Candidate selection queries refer to the problem of finding queries, for each instance s in a source set G_s, that retrieve a set of possible candidate matches for s in the target dataset G_t. Ideally, assuming that there is a 1-to-1 mapping between the instances, the best query produces candidates set with cardinality 1, because we can not have less than one candidate. We pose the problem as an optimization problem, where the goal is to find the most selective conjunction of query clauses that produces minimal candidate sets.

We focus on Web data including relational data, XML, RDF and other types of data that can be modeled as graphs $\mathbb{G}$ (e.g. RDF graphs), where every graph G $\in \mathbb{G}$ is a set of triples $(subject, predicate, object)$. Closely resembling this data model we employ a graph-structured data model:

Definition 1 (Data Model): Web data is are modeled as a set of graphs $\mathbb{G}$, where every graph G $\in \mathbb{G}$ is a set of triples, each of the form (s, p, o) where s $\in$U (called subject), p $\in$ U (predicate) and o $\in$ U $\cup$ L (object). Here, U denotes the set of Uniform Resource Identifiers (URIs) and L the set of literals, which together, form the vocabulary V, i.e. V=U $\cup$ L.

With respect to this model, instances are resources that appear at the subject position of triples. An instance representation can be obtained from the data graph as follows:

Definition 2 (Instance Representation): The instance representation $IR : \mathbb{G} \times U \rightarrow \mathbb{G}$ is a function, which given an instance $s \in U$ and a graph $G \in \mathbb{G}$, maps s to a set of triples in which s appears as the subject, i.e. $IR(s) = \{(s,p,o)|(s,p,o) \in G\}$.

We will use the terms instance and instance representation interchangeably from now on. Thus, only outgoing edges (s,p,o) of an instance s are considered. However, notice that the instance representation may be extended by including ingoing edges as well.

For a given set of source instances S, we can model a candidate selection query as a template that can be instantiate for each source instance s $\in$ S.

Definition 3 (Template Query): A template query Q is conjunction of n tuples (p,k), where p is a predicate in G_t and k is a token, defined as $\bigwedge_i^n (p_i,k_i) = \{t|(t,p_i,o) \in G_t \wedge o \sim k_i\}$. The similarity function $\sim$ is given. Without loss of generality we can also assume that can exist tuples where p are undefined, acting as a wild card.

Definition 4 (Candidate Set): A candidate set of an instance s in G_s is a instantiation of a template query Q where we have a tuple (p,k) where $(s,x,k) \in IR(s)$

In this work, we discuss how query templates can be derived. Also, executing all these queries is expensive, hence we propose to reduce the number of queries that has to be executed for retrieving candidates for every instance in the source.

Solutions. Aiming to approximate our solution to the optimal solution, we approach this problem in an iterative fashion, where at each iteration we use information from the previous iteration to refine and increase the selectivity of the query templates used on next iterations. Notice that as more selective a query is, as more efficient and effective it is, because, it selects less elements and it takes less time; and it selects less incorrect matches.

Without any knowledge of the source or target schemas, we start the process with queries templates that are less selective but easy to build. As the process moves on, information obtained from the candidate sets produced in previous iterations are used to refine the query templates for next iterations. Basically, this refining process adds two types of clauses in the query templates, aiming to make them more selective: attribute and class clauses. Attributes clauses are composed of highly selective target predicates with values similar to the values of at least one highly selective source predicate. The value of this source predicate is used as the object value of the attribute clause. To build attribute clauses we apply the Algorithm 2 described in our previous work [**?**] over the source instances and instances in the candidate sets already generated. Class clauses are predicate/value pairs that represent the class of interest of the target instances (e.g. rdfs:type=geo:country). To build class clauses, we apply a matcher over the already generated candidate sets obtaining positive and negative examples. Then, those positive and negatives examples are input to an algorithm that output the set of predicate/value pairs that select all positives and avoid the negatives examples; namely, the class of interest. Finally, those pairs are used to compose the class clauses. The matcher can be any approach that uses a more complex similarity measure to select the correct matches (positive examples) among the possible candidates. In this work, we assume that the source instances belong to a specific class of interest (e.g. countries), therefore we use the class-based disambiguation paradigm as the matcher. We do so, because this is the only complex matcher that can lead with datasets with non-overlapping schemas.

The set of query templates generated in this process can be large and their results for a specific instance may overlap. Hence, a set of heuristic is applied to decide whether or not to evaluate a template, aiming to avoid evaluating templates that will produce overlapping candidate sets. Those heuristics are embedded in a branch-and-bound optimization framework that on-line learns to efficiently evaluate the most effective query templates. As result, this framework produces the minimal candidate set passing only once over each source instance.

II. Conclusions