Towards explainable entity matching
via comparison queries

Alina Petrova, Egor V. Kostylev, Bernardo Cuenca Grau, and Ian Horrocks

Department of Computer Science, University of Oxford
{alina.petrova, egor.kostylev, bernardo.cuenca.grau, ian.horrocks}@cs.ox.ac.uk

Nowadays there exists an abundance of heterogeneous Semantic Web data coming from multiple sources. As a result, matching Linked Data has become a tedious and non-transparent task. One way to facilitate entity matching across datasets is to provide human-readable explanations that highlight what the two entities have in common, as well as what differentiates the two entities.

Entity comparison is an important information exploration problem that has recently gained considerable research attention [1, 2, 4]. In this paper we propose a solution towards explainable entity matching in Linked Data where entity comparison is used as a subroutine that assists in debugging and validation of matchings. To this end, we adopt the entity comparison framework in which explanations are modelled as unary conjunctive queries of restricted form [3, 4].

We concentrate on the data model where a dataset is an RDF graph—that is, a set of triples of IRIs and literals, jointly called entities. The basic building block of a query is a triple pattern, which is a triple of entities and variables. Then, a query is a non-empty finite set of triple patterns in which one variable, usually denoted by $X$, is an answer variable. The set $Q(D)$ of answer entities to a query $Q$ on a dataset $D$ is defined as usual in databases.

The two main notions of the framework are the similarity and difference queries for pairs of entities, which are defined as follows: a similarity query for entities $a$ and $b$ in a dataset $D$ is a query $Q$ satisfying $\{a, b\} \subseteq Q(D)$; a difference query for $a$ relative to $b$ is a query $Q$ satisfying $a \in Q(D)$ and $b \notin Q(D)$.

In our prior work we proposed an algorithm for computing comparison queries that can be repurposed for similarity and difference queries [3]. The algorithm is based on the computation of a similarity tree—a data structure that represents commonalities and discrepancies in data for input entities $a$ and $b$. It is a directed rooted tree with nodes and edges labelled by pairs of sets of entities such that the root is labelled by $(\{a\}, \{b\})$ and every edge labelled $(E_1, E_2)$ between nodes labelled $(N_1, N_2)$ and $(N'_1, N'_2)$ is justified in the sense that for every entity $n$ in $N_i$, $i \in \{1, 2\}$, there is a triple $(n, e, n')$ in the dataset with $e \in E_i$ and $n' \in N'_i$.

For instance, suppose there are 3 entities, Emma Watson, Emily Watson and E. Watson, that need to be either matched or disambiguated, and a data fragment given in Figure 1. Then the similarity trees for Emma Watson and E. Watson, and for Emily Watson and E. Watson are depicted in Figure 2 (where singleton sets $\{\ell\}$ and pairs $(\{\ell\}, \{\ell\})$ are both written as $\ell$ for readability).
Each branch in a similarity tree can be treated as a separate similarity query, in which each edge is encoded as a triple pattern, and each label \((L_1, L_2)\) is encoded as either an entity \(\ell\) if \(L_1 = L_2 = \{\ell\}\) or a fresh variable otherwise. For example, a query \(Q_1 = (X, \text{actedIn}, \text{Ballet\_Shoes})\) is a similarity query for \(\text{Emma\_Watson}\) and \(\text{E\_Watson}\), while a query \(Q_2 = (X, \text{actedIn}, Y), (Y, \text{year}, Z)\) is a similarity query for \(\text{Emily\_Watson}\) and \(\text{E\_Watson}\). Moreover, each branch involving non-entity labels can also be treated as a difference query, if instead of some variables we take entities from one of the label sets. For example, query

\[
Q_3 = (X, \text{actedIn}, \text{Little\_Women}), (\text{Little\_Women}, \text{year}, 2019)
\]

is a difference query for \(\text{E\_Watson}\) relative to \(\text{Emily\_Watson}\).

Both types of queries can assist in explaining why two entities should or should not be merged: \(Q_1\) gives a good reason to match \(\text{Emma\_Watson}\) and \(\text{E\_Watson}\) into one entity, \(Q_2\) is not specific enough to match the other pair, and \(Q_3\) can act as an indicator that the two movies named \(\text{Little\_Women}\) are indeed two different movies, and \(\text{Emily\_Watson}\) and \(\text{E\_Watson}\) are different people.

**References**