MetKB: Enriching RDF Knowledge Bases with Web Entity-Attribute Tables

Haoqiong Bian, Yueguo Chen*, Xiaoyong Du, Xiaolu Zhang
Key Laboratory of Data Engineering and Knowledge Engineering (Renmin University of China), MOE, China
School of Information, Renmin University of China, Beijing, China
{bianhq,chenyueguo,duyong,zxlruc2010}@ruc.edu.cn

ABSTRACT

There are many entity-attribute tables on the Web that can be utilized for enriching the entities of an RDF knowledge base. This requires the schema mapping (matching) between the Web tables and the RDF knowledge base. In this paper, we propose a feasible solution that is able to automatically search and rank entity-attribute tables from the Web, and effectively map the extracted tables with the RDF knowledge base with very few manual efforts.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Web table, RDF knowledge base, schema mapping

1. INTRODUCTION

There are many Web-scale knowledge bases (KBs) such as FreeBase, YAGO, Linked Data, which utilize RDF triples to represent their basic information units. They have been widely used in applications such as semantic search, text understanding and question answering [2]. To effectively support these applications, an RDF KB needs have information of a huge number of open domain entities. Although the size of Web-scale KBs keeps growing very fast, the coverage of a single KB is still very limited, compared to the numerous entities in the real world.

The current Web contains billions of tables, among which a huge number of tables (154M found in the Webtables project [1]) contain high-quality relational data. Of these high qualified tables, there are many entity-attribute tables that contain information of some entities of the same type [1, 3]. In such tables, information of an entity appears in one row with each column representing an attribute of the entity. It is possible that some entities in a Web table may have corresponding entries in the KBs, from which we may learn the mapping between the Web table and the KBs. With the mapping, we are able to automatically inject entities of the Web table into the KBs. The key problem is therefore a schema mapping problem in which we want to find the mapping between a Web table and a KB.

We have proposed techniques for efficiently and effectively conduct schema mapping between a Web table and an RDF KB [4]. However, finding proper entity-attribute tables from the Web for enriching RDF KBs remains a challenge. The reasons are at least in three folds: 1) although there are many tables on the Web, many of them may not be entity-attribute tables; 2) even for an entity-attribute table, it may not be beneficial for the entity enriching task if there are few novel entities or the table cannot be integrated with the RDF KB; 3) the schema mapping automatically detected between a Web table and the RDF KB is often error-prone, some manual efforts to validate the schema mapping are necessary to guarantee the quality of KB enlargement.

In this paper, we introduce a system, called MetKB, which is able to automatically search and rank entity-attribute tables from the Web, and effectively map the extracted tables with the RDF KB with very few manual efforts. It always suggests best schema mappings detected so far for users' validation, so that we can populate and enrich the entities of a KB with high quality guaranteed.

2. OVERVIEW

The MetKB system is designed to address the following challenges:

- How to efficiently and effectively find relevant entity-attribute tables from the Web?
- How to evaluate the feasibility and the value of integrating an entity-attribute table to an RDF KB?
- How to guarantee the quality of entities extracted from the entity-attribute table and injected to the KB?

Figure 1 illustrates the solution of the MetKB system, which consists of several main steps (in the clockwise order). Among them, there are two main steps that involve human intervention. One is the initial step in which users pick an entity type that requires to populate entities. The other one is the validating step that continuously feeds the best schema mapping detected so far for users' feedbacks and validation.

*Contact author, and the authors are supported by National Basic Research Program of China (973 Program) No. 2012CB316205, and the National Science Foundation of China under grant No. 61170010 and 61003085.

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.
ACM 978-1-4503-2263-8/13/10.
http://dx.doi.org/10.1145/2505515.2508209.

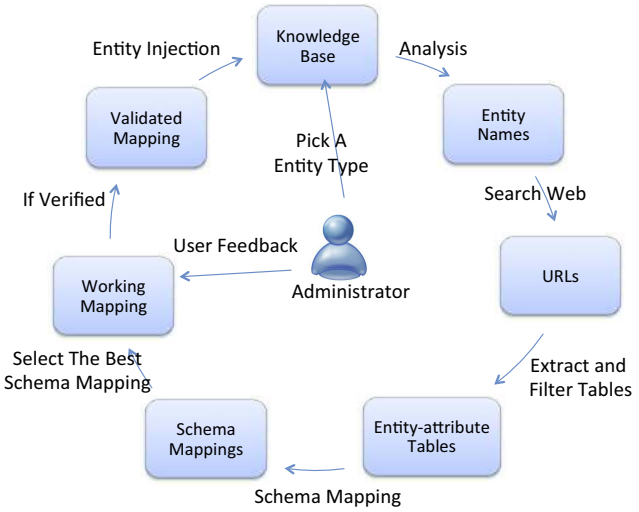

Figure 1: The information flow of MetKB

2.1 Picking an Entity Type

The system provides a list of entity types for enlarging entity population. Once the user selects an entity type, a whole process of populating entities will be triggered. The system will load information of entities of the picked type into memory. The entities will be ranked based on their novelty (timestamps when the entities are injected into the KB) and their seeding ability (evaluated in Sec. 2.2). Entities of higher novelties and higher seeding abilities will be ranked higher, and therefore be processed earlier. When the type is picked in the first time, entities of the picked type will be ranked only based on their seeding ability (computed offline). However, with more and more novel entities are inserted into the KB, the novelty will be weighted more when ranking entities for searching Web tables.

2.2 Searching Web Tables

This step continuously consumes the top entity of the entity list generated from the previous step. For each entity, it generates a Web table search query using the entity name (quoted) and the picked entity type as query keywords. The query is submitted to the Google Web Table search engine[1] from which we are usually able to retrieve Web tables containing the given entity name. From the snippets of the search results, we can directly know the size of the retrieved table. We then roughly ignore search results whose table size is not large enough. For a query, the utility of a search result T is evaluated as $u(T) = \frac{\sqrt{size(T)}}{rank(T)}$, where $size(T)$ is the number of cells of T, and $rank(T)$ is the rank of T in the search results of the corresponding query. The seeding ability of an entity is then evaluated as the sum of utilities of top-k search results whose size is large enough. In our study, we straightforwardly set $k = 20$.

Note that, the same table can be retrieved by different queries. The overall utility of the table will be the sum of its utilities over all the search results. As entities are continuously consumed and fed to the search engine, a large number of Web tables will be retrieved. They are dynamically ranked based on their overall utilities generated from a series of queries.

[1]http://research.google.com/tables

2.3 Extracting and Filtering Web Tables

This step is the consumer of the ranking list of Web tables generated from the previous step. As long as the Web table list is long enough, it continuously consumes the top Web table from the list. For each Web table (which is actually a URL so far), a table extraction and filtering process will be conducted. The applied table extractor is quite simple. It extracts tables only from shallow Web pages, in which tables are discovered by HTML table tags. The table headers are ignored in our study (note that some tables do not contain headers). Nested tables (e.g., a cell contain multiple sub-cells, or multiple cells are merged together) are also ignored in our study for guaranteeing the quality of the extracted information. Finally, Web tables are transferred into plain tables where the information in each cell is extracted (unless it is a vacant cell).

An entity-attribute table contains information of entities. We define the column containing entity names as the key column of a table. It satisfies some constraints: 1) no duplicated names in the column; 2) its values are not numerical or IDs. This is reasonable because one of our assumptions is that the applied entity-attribute Web table should contain information of entities distinguished by their names within a column. If a table has more than one candidate key columns, only the first one will be picked as the key column. Note that we ignore tables whose entity names are distributed over multiple columns. Fig.2 shows an example of an entity-attribute table and its key column. A table will be filtered if it does not contain a key column. If a page contains multiple tables with key columns, only the one with the largest number of rows will be picked for further processing. In order to find good schema mappings, we filter columns containing numerical IDs or too many vacant values.

1	Grand Illusion	true	Jean Renoir	France	1937
2	Seven Samurai	true	Akira Kurosawa	Japan	1954
3	The Lady Vanishes	true	Alfred Hitchcock	United Kingdom	1938
4	Amarcord	true	Federico Fellini	Italy	1974
5	The 400 Blows	true	François Truffaut	France	1959
6	The Naked Kiss	false	Samuel Fuller	United States	1964
7	Shock Corridor	false	Samuel Fuller	United States	1963

Figure 2: A key column of a Web table

Note that most non entity-attribute tables will be filtered in this step. Even for an entity-attribute table, some columns can also be filtered from the original tables. Finally, the output of the table extraction and filtering is a list of prepared tables that are ready for schema mapping. In this step, we will match the discovered entity names to those existing entity names of a given entity type in KB. The utility of a prepared table T is evaluated as $u(T) = m(T)n(T)c(T)$, where $m(T)$ is the number of entity names in T that exists in the KB, $n(T)$ is the number of novel entity names in T that does not exist in the KB, and $c(T)$ is the number of columns of T. With this, we rank all the prepared tables for further processing.

2.4 Schema Mapping

This step is the consumer of the ranking list of prepared tables generated from the previous step. Each time, it processes a prepared table of the largest utility, based on the

schema mapping techniques proposed in [4]. As the target of the schema mapping has been fixed as the mapping between a prepared table and a selected entity type, we only generate the best schema mapping for each schema mapping task. Considering that the information of entities of the selected type has been loaded into memory in the first step, the mapping task can therefore be conducted very efficiently. The utility of a mapping has been defined in [4], from which we are able to rank all the schema mappings detected between prepared tables and the given entity type. Details of how schema mapping is efficiently and effectively conducted are given in the reference [4].

2.5 Verifying The Mapping

To guarantee the quality of KB enlargement, the detected schema mappings need to be manually verified before they can be used for injecting novel entities from tables to the KB. A user can check whether a column of the table is correctly matched to a predicate of the KBs. The user can modify the mapping by removing/modifying those incorrect mappings between columns and predicates. For those columns that cannot be automatically matched to any predicate, the user may help to pick a predicate from the predicate list of the given entity type, if he is sure that there is a correct mapping between the picked predicate and the corresponding column.

To effectively illustrate a schema mapping, examples of mapped tuples will be highlighted. A user can choose to reject a schema mapping of a table if he is not satisfied with the detected results. Otherwise, information of the table will be utilized for entity enlargement.

2.6 Entity Injection

Once a mapping is verified, the system will inject information of entities in the table into the KB. The values in the key column will be transformed into the subjects of entities. Those in the other mapping columns are treated as objects. The predicates are directly derived from the mapping. Note that information in entity-attribute tables does not contain prefixes that are widely used in RDF KBs. However, the prefixes of mapping predicates are given in the KB. We only need to assign prefixes to subjects and objects. The prefixes can be learned by majority voting through those detected matched entities in the KB. For objects, we will not assign prefixes if the learned prefixes are not confident enough. Along with the injection of entities, the indexes of entities will be updated accordingly for further schema mapping of the selected entity type. The insertions will be logged to support the rollback of entity injection, in case that some verified schema mappings or information of the source tables are not correct.

3. DEMONSTRATION

We use the DBPedia 3.7 dataset as the underlying RDF KB in the current MetKB system [2]. The DBPedia RDF KB contains 3.64 millions of entities. The most popular types of entities in the KB include persons, places, music albums, films, etc. The system is implemented as a working prototype written in Java, offering a Web-based interface to allow users to select entity types for population enlargement and verifying the discovered schema mappings. In the meanwhile, the interface provides information for monitoring the status of intermediate processes of the system.

For the demonstration, we will allow a user to pick an entity type from a list of common entity types existing in the KB. After that, the user need to wait for dozens of seconds, during which the MetKB system sends queries to the Google Web Table search engine for retrieving URLs containing the relevant Web tables, extracts and filters Web tables, and integrates the prepared tables with entities of the picked type. After finding a certain number of schema mappings, the system will automatically push the best schema mapping to the user for his feedbacks and verification.

For a schema mapping shown out for user's feedbacks, we will explain why the table (a link to the original Web page containing the table will be given) is mapped to the existing entities of the given type, based on some highlighted information (cells that have mapping entries to a mapping entity in the KB) in the Web portal. We will allow the user to modify the schema mapping, e.g., by replacing the mapping predicate of a column with some others, by removing the mapping of a column, or by providing a mapping predicate suggested by the system to an unmatched column. Based on the mapping results, the users can also either approve a schema mapping or reject it directly. Once a schema mapping is approved (verified), novel information of the table will be automatically injected into the KB. After a schema mapping has been processed by the user (either approved or rejected), the system will automatically suggest the next best schema mapping to the user for further feedbacks.

The system provides many statistics to allow users to monitor the status of the whole loop of KB enlargement. For example, it shows the total number of entities of the picked entity type existing in the KB, the number of novel entities discovered so far from the Web tables, the ratio of discovered schema mappings that are verified by the user (it often helps the user to judge whether to terminate the KB enlargement process or not), the number of Web pages that have been crawled, the number of entities that have been used for search Web tables, the number of entries in each ranking list, etc.

The system allow administrators for adjusting parameters (using a configure file) used for schema mapping, through which the users are able to experience the impacts of these parameters on the Web tables that can be discovered by the system. We have tested the system using some common types of entities in the applied RDF KB. The results show that for most of these types, our system can effectively discover a large number of high-qualify Web tables for enriching entities of the KB.

[2] http://202.112.114.27:8080/MetKB

4. REFERENCES

[1] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. *PVLDB*, 1(1):538–549, 2008.

[2] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD Conference*, pages 481–492, 2012.

[3] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *SIGMOD Conference*, pages 97–108, 2012.

[4] X. Zhang, Y. Chen, J. Chen, X. Du, and L. Zou. Mapping entity-attribute web tables to web-scale knowledge bases. In *DASFAA(2)*, pages 108–122, 2013.