Web Semantics: Science, Services and Agents on the World Wide Web 35 (2015) 159–166

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

The Mannheim Search Join Engine

Oliver Lehmberg *, Dominique Ritze, Petar Ristoski, Robert Meusel, Heiko Paulheim, Christian Bizer

Data and Web Science Group, University of Mannheim, B6, 26, 68159 Mannheim, Germany

ARTICLE INFO

Article history:
Received 30 January 2015
Accepted 1 May 2015
Available online 30 May 2015

Keywords:
Table extension
Data search
Search Joins
Web tables
Microdata
Linked data

ABSTRACT

A Search Join is a join operation which extends a user-provided table with additional attributes based on a large corpus of heterogeneous data originating from the Web or corporate intranets. Search Joins are useful within a wide range of application scenarios: Imagine you are an analyst having a local table describing companies and you want to extend this table with attributes containing the headquarters, turnover, and revenue of each company. Or imagine you are a film enthusiast and want to extend a table describing films with attributes like director, genre, and release date of each film. This article presents the *Mannheim Search Join Engine* which automatically performs such table extension operations based on a large corpus of Web data. Given a local table, the Mannheim Search Join Engine searches the corpus for additional data describing the entities contained in the input table. The discovered data are joined with the local table and are consolidated using schema matching and data fusion techniques. As a result, the user is presented with an extended table and given the opportunity to examine the provenance of the added data. We evaluate the Mannheim Search Join Engine using heterogeneous data originating from over one million different websites. The data corpus consists of HTML tables, as well as Linked Data and Microdata annotations which are converted into tabular form. Our experiments show that the Mannheim Search Join Engine achieves a coverage close to 100% and a precision of around 90% for the tasks of extending tables describing cities, companies, countries, drugs, books, films, and songs.

1. Introduction

Imagine you are a marketing manager who wants to group the customers of a company according to different properties of the countries in which the customers are located in order to select customers that should be targeted by a marketing campaign. While the data about customers can be found in the company's internal data sources, further background information about the countries may not be stored there. Relevant data about countries could for instance include population, area, GDP, or human development index. Today, the manager needs to manually search and integrate data about each country using search engines such as Google, access the small set of online databases he knows about, or copy-and-paste values from Wikipedia. Manually searching for data is cumbersome and the manager will likely miss a large fraction of all relevant data sources that are available on the Web.

This article presents the *Mannheim Search Join Engine (MSJ Engine)* which supports the manager in reaching his goal by automating the data search and integration tasks. Given a local table, the MSJ Engine searches a heterogeneous data corpus for additional data describing the entities contained in the input table. The search operation does not assume any external knowledge about correspondences on schema or instance level. The discovered data are joined with the local table and their content is consolidated using schema matching and data fusion methods. As a result, the user is presented with an extended table and given the opportunity to examine the provenance of the added data. We evaluate the engine using a data corpus consisting of 36 million tables originating from over one million different websites. The data corpus contains HTML tables as well as Linked Data [1] and Microdata annotations [2] which are converted into tabular form. In contrast to the existing research on table extension by Google [3] and Microsoft [4], our data corpus as well as the source code of the MSJ Engine are publicly available.

The article is structured as follows: Section 2 gives an overview of the architecture of the MSJ Engine and describes the methods

* Corresponding author.
E-mail addresses: oli@informatik.uni-mannheim.de (O. Lehmberg), dominique@informatik.uni-mannheim.de (D. Ritze), petar.ristoski@informatik.uni-mannheim.de (P. Ristoski), robert@informatik.uni-mannheim.de (R. Meusel), heiko@informatik.uni-mannheim.de (H. Paulheim), chris@informatik.uni-mannheim.de (C. Bizer).

http://dx.doi.org/10.1016/j.websem.2015.05.001
1570-8268/

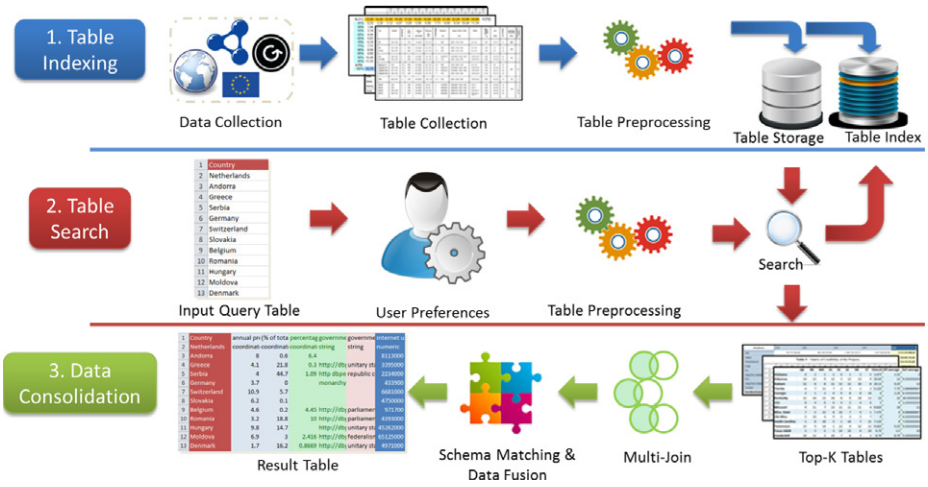

Fig. 1. Functionality of the MSJ Engine.

that are employed for data normalization, data search, and data consolidation. Section 3 describes the data corpus that was used to evaluate the engine and presents the results of our table extension experiments. Section 4 compares the MSJ Engine with related work, while Section 5 outlines directions for future work.

2. Architecture

A Search Join is a join operation which extends a local, user-provided table (called query table) with additional attributes based on a large corpus of heterogeneous tabular data [5]. Search Joins can be described as a concatenation of three operations: a search operation s, a multi-join operation m, and a consolidation operation c:

$$R = c(m(T_q, s_{T_q,a}(\mathbb{T}_c))) \quad (1)$$

where R is the result table, T_q is the query table, $\mathbb{T}_c$ is a corpus of heterogeneous tables, and a is an optional, user-specified parameter that allows the search to be constrained to a specific attribute (constrained query), e.g. only search for information about the population of countries. If the parameter a is not specified, all discovered attributes will be added to the result table (unconstrained query).

Fig. 1 gives an overview of the functionality of the MSJ Engine. The functionality can be divided into three areas: Table indexing, table search, and data consolidation. In the following, we will describe each area in detail.

2.1. Table indexing

The MSJ Engine uses simple entity–attribute tables as internal data model. Each table describes a set of entities using a set of single-valued attributes. Each attribute has a header which we assume to be some surface form of the attribute's semantic intention. We distinguish the following attribute data types: nominal values, numbers with or without unit of measurement, timestamps, and geo-coordinates. As many Web and intranet data sources provide natural language labels for the entities that they describe, we require each table to contain a *subject attribute* containing the entity name, e.g. Lady Gaga, U.S.A. or United States. This approach to rely on entity names as pseudo-keys is also used by the related work from Google [3,6] and Microsoft [4].

Before a table from the data corpus $\mathbb{T}_c$ is indexed, the value of each cell is normalized, i.e., tokenized, lower cased, values in brackets and stop words are removed.

In order to be indexed, a table needs to fulfill two conditions: (1) it must contain at least three attributes and describe at least five entities, (2) it must contain a subject attribute. Applying these filtering conditions ensures a minimal quality of the remaining tables.

We apply the following heuristic in order to identify the subject attribute of a table: If a table contains an *rdfs:label* attribute or some other attribute having a header containing the string *name*, this attribute will be chosen as subject attribute. Otherwise, the string attribute with the highest number of unique values is chosen as subject attribute. In cases where two or more attributes contain equally high numbers of unique values, the left-most attribute is chosen. Furthermore, we only consider attributes with at least 60% unique values.

To identify attribute headers, we use the following heuristic: We assume that the attribute headers are in the first non-empty row of the Web table that contains at least 80% non-empty cells. We present an evaluation of both pre-processing heuristics in Section 3.2.

The data type of each table attribute is identified based on its values. First, the data type of each value of the attribute is detected by using about 100 manually defined regular expressions which are able to detect the data types number (with or without unit of measurement), timestamp and geo-coordinates. Additionally, the algorithm uses around 200 manually generated rules for converting units of measurements to the corresponding base unit (metric system), e.g. *8 sq. mi.* will be converted to 20.72 *million square meter*. After the data type of each value is detected, the final attribute data type is decided using majority voting.

As a final step of the indexing procedure, the normalized subject attribute values and attribute headers of each table are stored in a *Lucene index*,[1] which is used later by the search operation.

2.2. Table search

The query table T_q is preprocessed in the same manner as the tables of the data corpus $\mathbb{T}_c$. Then, the search operator s is applied and tries to find matching subject values in the previously indexed tables $\mathbb{T}_c$. For deciding whether a subject value from a table matches a subject value from the query table, two different methods are available: exact subject value matching, and similar subject value matching.

[1] http://lucene.apache.org/.