Keyword Search over Data Service Integration for Accurate Results

Vidmantas Zemleris[1], Valentin Y Kuznetsov[2] and Peter Kreuzer[3]

[1]Faculty of Mathematics and Informatics, Vilnius University, Lithuania and CMS Experiment at CERN, Geneva, Switzerland
[2]Cornell University, USA
[3]Rheinisch-Westfaelische Tech. Hoch., Germany

E-mail: [1]vidmantas.zemleris@cern.ch

Abstract. Virtual data integration aims at providing a coherent interface for querying heterogeneous data sources (e.g., web services, web forms, proprietary systems) with minimum upfront integration effort.

Querying is usually carried out through a structured query language, such as SQL, which forces the users to learn the language and to get acquainted with data organization (i.e. the schema) thus negatively impacting the system's usability.

We present a keyword search system which deals with this problem by operating on available information: the metadata, such as the constraints on allowed values, analysis of user queries, and certain portions of data. Given a keyword query, it proposes a ranked list of structured queries along with the explanations of their meanings. Unlike previous implementations, the system is freely available and makes no assumptions about the input query, while maintaining its ability to leverage the query's structural patterns - in case they exist. The system is discussed in the context of CMS data discovery service where the simplicity and capabilities of the search interface play a crucial role in the ability of its users to satisfy their information needs.

1. Introduction

In *Virtual Data Integration* (EII), data physically stays at its origin, and is requested only on demand, usually, through structured query languages such as *SQL*. To allow querying the sources in a coherent way, EII performs a number of transformations on queries and their results (eliminating the inconsistencies in data formats and [entity] naming; combine the results, etc.). Unfortunately, it requires the users to learn the query language and to get acquainted with organization of data (i.e. the mediated schema), which is often not straight-forward. Virtual integration presents an additional challenge, since only limited access to the data instances is available, rendering the traditional keyword search methods that return data-tuples inapplicable (e.g. Information Retrieval or Keyword search over relational databases).

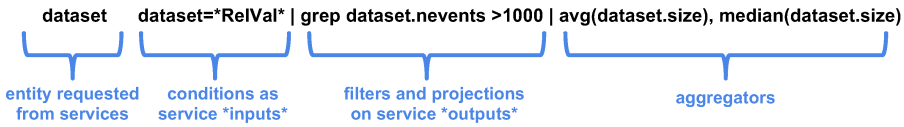
The objective of this work is to investigate keyword search that proposes a ranked list of structured queries, as a more intuitive alternative, which, in fact, received relatively little attention in the field of data integration[1]. The keyword search implementation that we present is being developed as part of the "DAS" virtual service integration system used at the CMS Experiment, CERN and cover some specific use-cases there, however is mostly generic and could be used with other EII system only with minor adjustments.

2. Preliminaries

2.1. DAS - a system for virtual service integration

"CMS Data Aggregation System" *(DAS)* [2, 3] provides an integrated access to a number of proprietary data-sources through simple structured queries eliminating the inconsistencies in entity naming, data formats and combining the results.

Its queries are formed specifying the entity the user is interested in (e.g. dataset, file, etc) and providing selection criteria (e.g. attribute=value, attribute *between* [v1, v2]). The results could be later 'piped' for further filtering, sorting or aggregation (min, max, avg, sum, etc.), e.g.:



As seen above, DASQL closely corresponds to the physical execution flow: based on the requested entity and the conditions on service inputs, DAS decides the set of services to be queried[1]. Then, after retrieving, processing and merging the results from services, the filters and projections, and aggregators are applied. The results are cached for subsequent uses.

2.2. Problem definition

Given a keyword query, KWQ=$(kw_1, kw_2, .., kw_n)$, we are interested in translating it into the corresponding structured query. To do so we are *interpreting* it in terms of its semantics over the *integration schema* which is also given:

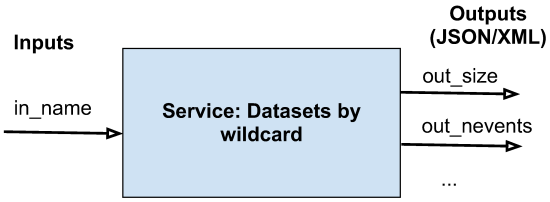


Figure 1. a data-service (simplified)

- *schema terms*: names of entities and their attributes (either *inputs* to the services or their *output* fields)
- information about possible *value terms*:
  - for some fields, a list of possible values
  - *constraints* on allowed as data-service inputs (required fields, regular expressions defining the values accepted)

Consider these queries: "*what is the average size of RelVal datasets where number of events is more than 1000*", "*avg dataset size Zmmg number of events>1000*" and "*avg(dataset size) Zmmg 'number of events'>1000*". For all, the expected result is:

In the particular case of DASQL, used at CMS, the input has to be mapped into:

- type of result entity (e.g. datasets) and projections of fields in the service outputs
- conditions that will be passed to services as their inputs, e.g. datset=*RelVal*
- post-filters on service outputs: e.g. dataset.nevents > 1000
- basic aggregation functions, applied on service results: e.g. avg(dataset.size)

[1] including pre-defined "virtual services", which feed results from one service into inputs of the others

2.3. State of the art

Nature of keyword queries Keyword queries are often underspecified, therefore every possible interpretation has to be included in the results [4]. Still, some interpretations are more likely than the others, therefore, when the users are interested in complete answer sets, the standard approach is to produce a ranked list of most-likely structured queries [5, 6, 7].

Further, it has been noticed that even if keyword queries do not have any clear syntactic structure, keywords referring to related concepts usually come close to each other in the query [8, 4]. Most of the existing approaches attempt to profit from these dependencies to ameliorate their candidate answers ranking.

3. From Keywords to Queries

3.1. Overview

3.2. Tokenization

3.3. Scoring individual keywords (entry points)

3.4. Ranking function

4. Generating the results

4.1. via Exhaustive search

allows filtering only those that DIS supports.

4.2. via Solving weighted-bipartite assignments

4.3. Combination(?)

exhaustive for part that is well

5. Our implementation

cython

6. Future work

Terminology

Data Virtualization...

7. Related work

Keyword querying over EII Two approaches were identified as the closest to our problem:

Keymantic [9, 6] answers keyword queries over relational databases with limited access to the data instances or over data integration [6]. First, based on meta-data[2], individual keywords are scored as potential matches to *schema terms* (entity names and their attributes, using some entity matching techniques) or as potential *value* matches (by checking any available constraints, such as the regular expressions imposed by the database or data-services). Then, these scores are combined**,** and refined by heuristics that increase the scores of query interpretations with the nearby keywords having related labels assigned. Finally, these labels are interpreted as SQL queries.

KEYRY [7] attempts to incorporate users feedback through training an Hidden Markov Model's (HMM) tagger taking keywords as its input. It uses the List-Viterbi [10] algorithm to produce the top-k most probable tagging sequences (where tags represent the "meaning" of each keyword). This is interpreted as SQL queries and presented to the users. The HMM is first initialized through the supervised training, but even if no training data is available, the initial HMM probability distributions can be estimated through a number of heuristic rules (e.g. promoting related tags). Later, user's feedback can be used for supervised training, while even

[2] in EII, only limited access to data instances is available, therefore instead of just indexing the all data, the meta-data shall be used

the keyword queries itself can serve for unsupervised training [11]. According to [7] the accuracy of the two systems didn't differ much.

8. Conclusions

References

[1] Guerrisi V, La Torre P and Quarteroni S 2012 *Search Computing* 82–97

[2] Kuznetsov V, Evans D and Metson S 2010 *Procedia Computer Science* **1** 1535 – 1543 ISSN 1877-0509 iCCS 2010 URL http://www.sciencedirect.com/science/article/pii/S1877050910001730

[3] Ball G, Kuznetsov V, Evans D and Metson S 2011 *Journal of Physics: Conference Series* **331** 042029 URL http://stacks.iop.org/1742-6596/331/i=4/a=042029

[4] Bergamaschi S, Domnori E, Guerra F, Lado R T and Velegrakis Y 2011 Keyword search over relational databases: a metadata approach *Proceedings of the 2011 international conference on Management of data* (ACM) pp 565–576

[5] Blunschi L, Jossen C, Kossmann D, Mori M and Stockinger K 2012 *Proc. VLDB Endow.* **5** 932–943 ISSN 2150-8097 URL http://dl.acm.org/citation.cfm?id=2336664.2336667

[6] Bergamaschi S, Domnori E, Guerra F, Orsini M, Lado R T and Velegrakis Y 2010 *Proc. VLDB Endow.* **3** 1637–1640 ISSN 2150-8097 URL http://dl.acm.org/citation.cfm?id=1920841.1921059

[7] Bergamaschi S, Guerra F, Rota S and Velegrakis Y 2011 *Conceptual Modeling–ER 2011* 411–420

[8] Kumar R and Tomkins A 2009 *IEEE Data Engineering Bulletin* **32** 3–11

[9] Bergamaschi S, Domnori E, Guerra F, Trillo Lado R and Velegrakis Y 2011 Keyword search over relational databases: a metadata approach *Proceedings of the 2011 international conference on Management of data* (ACM) pp 565–576 URL http://dl.acm.org/citation.cfm?id=1989383

[10] Seshadri N and Sundberg C 1994 *Communications, IEEE Transactions on* **42** 313–323

[11] Rota S, Bergamaschi S and Guerra F 2011 The list viterbi training algorithm and its application to keyword search over databases *Proceedings of the 20th ACM international conference on Information and knowledge management* (ACM) pp 1601–1606