Ontology Based Data Access Methods to Teach Students to Transform Traditional Information Systems and Simplify Decision Making Process

Svetlana Chuprina¹, Igor Postanogov¹, and Olfa Nasraoui²

¹ Perm State University, Bukireva Str. 15, 614990, Perm, Russia  
chuprinas@inbox.ru, ipostanogov@outlook.com

² Knowledge Discovery & Web Mining Lab, University of Louisville, Louisville KY 40292, USA  
olfa.nasraoui@louisville.edu

Abstract

We describe a service-based approach that provides a natural language interface to legacy information systems, built on top of relational database management systems. The long term goal is to make data management and analysis accessible to a wider range of users for a diverse range of purposes and to simplify the decision making process. We present an ontology-driven web-service, named Reply, that transforms traditional information systems into intelligent systems, endowed with a natural language interface, so that they can be queried by any novice user much like modern day search engines. The principal mechanism of our approach is turning a natural language query into a SQL-query for structured data sources by using Ontology-Based Data Access methods. We also outline how the proposed approach allows semantic searching of large structured, unstructured, or semi-structured data within the database or outside sources, thus helping bridge the talent gap in the case of Big Data Analytics used by researchers and postgraduate students.

Keywords: Bridging the Talent Gap in Data Analytics, Legacy Information System, Intelligent Information System, Natural Language Interface, Ontology-Driven System, Ontology-Based Data Access, Semantic Web, Open Data

1 Introduction and Motivation

Until recently, data analysis was limited to the purview of a small community of IT-specialists. In recent years, however, it has become of an interest to the general public. Making data analysis useful to a wider range of specialists, not only in the IT area, but also in economics, ecology, medicine and other areas, as well as casual users, for a diverse range of purposes, is one of the key challenges to be addressed in the development of data-driven systems, especially due to the increasing use of Big Data analytics tools.
In order to address the talent gap in natural sciences and high performance computing (HPC), Decision Making Tools must become more and more accessible to the general user. One of the urgent tasks that can contribute toward this goal is to enrich existing information systems with a natural language (NL) interface, allowing the non-expert or the beginner user to ask questions in natural language to query structured, unstructured and semi-structured data in a uniform way.

This paper aims to describe our methodological approach to training students in the Masters Program “Applied Mathematics and Computer Science” at the Faculty of Mechanics and Mathematics, in Perm State University, Russia. We propose a new approach that helps students to both become acquainted with new methods of Big Data Analytics and also take part in their implementation. The main enabling mechanism in our approach is turning a natural language query into a SQL-query that can be submitted to structured data sources in legacy information systems (IS). It is assumed that the students are already familiar with database theory, have good skills in working with relational databases, and are familiar with the basics of ontology engineering methods.

It is well-known that the goals and needs of existing information systems evolve and expand over time. Hence, there is a need for unified programming tools, which optimize the automated modification of existing IS. We focus on situations in which the replacement of an existing IS by a new one is not viable due to high costs of new systems or other reasons such as the complexity and risks of data migration and needed re-configuration and development.

One of the most needed enhancements to legacy information system is supporting new types of queries. Straightforward solutions may result in existing source code modifications which will lead to well-known consequences. This task could be simplified if there could be a high level constructor of ad-hoc queries which requires only a modification in the query parser and structured query generator during the IS lifetime.

One of the most time and resource-consuming tasks is enriching a legacy IS with new NL query interface support. Many recent publications and business reports, devoted to human machine interfaces, have pointed to the crucial need of extending a legacy IS with a NL interface to relational databases so that they can be queried much like modern day search engines (here, we refer to an IS with RDB storage subsystems as a "traditional information system"). For example, there is a growing need for reusing publicly available data sources, as part of the open data ecosystem, and bringing them closer to the wider range of Big Data analytics, data science pipelines, and casual users.

It is not a coincidence, therefore, that NL interfaces are becoming increasingly popular. A NL interface lowers the entry threshold for new users, and also the training costs. Meanwhile however, there are considerable technological, social and psychological problems preventing the mass distribution of NL interfaces. The main problem is related to the ambiguity of NL-query interpretations, shallow analysis of a semantic query context and a talent gap in the NL processing query development. Over and underestimations of NL-query interpretation capabilities are widespread psychological problems. While some users are making queries with “all the bells and whistles” of NL, which are typically hard for machine interpretation, other users are still using overly simple keyword queries.

Additional problems arise when legacy systems are being upgraded. Legacy systems are systems that do not satisfy most modern requirements but are still very useful. We mention some big difficulties in their replacement: for example, it is hard to find an educational institution or corporation with more than 20 years of history, which does not use such an IS. The databases of these systems contain massive amounts of valuable data that organizations cannot work without. Shutting down the IS even for a short time is not even an option. On the other hand, it is really hard to maintain legacy systems because they are neither "open" (like open systems) nor at least informationally interoperable.

If an information system is used in a huge organization, even a simple expansion of supported structured queries may encounter administrative difficulties, lack of human resources, manpower turnover, and significant downtime for those to whom these new requirements are necessary. Also, there may be reasons why the changes to the source code of a legacy IS may be undesirable or impossible (due to the presence of only the executable code of the system). The study (Kharlamov et al., 2014)
showed that at Siemens Energy, analysts spend 80% of their time in formulating queries for finding relevant data. If at the level of the external interface of the IS, such query formulation is impossible, then a request must be made to the IT-branch. As noted, further refinement of the system is complicated due to the IT-specialist workload and the problems associated with the lack of understanding of tasks.

Apart from the problem that most of the legacy systems are monolithic, the use of obsolete programming languages, and lack of funds to support interoperability, the inability to make changes in the IS may have legal reasons (such as not being allowed to make changes in third-party components) or may be caused by inaccessibility or loss of source code of the software system. Using the proposed service-oriented approach of enhancing a legacy IS with a NL query interface, without any change to the source code driving the legacy system, can help tackle this problem.

Our approach is based on the methods of ontology engineering and could be uniformly applied to structured, semi-structured and unstructured information resources, as well as while dealing with distributed queries in Big Data. As an illustration of the applicability of the proposed approach, we use it to teach Masters students to transform traditional information systems into smart information systems with a NL interface and to develop web-services to simplify the decision making process.

2 Proposed Approach

NL question answering was at the peak of inflated expectations according to the Gartner Hype Cycle 2014 research report. Siri (iOS), Google Now (Android) and Cortana (Windows) are also examples of NL question answering systems in which text interpretation is preceded by voice recognition.

Querying Internet search engines using NL has been a standard for more than a decade. However, applying the same approach to traditional IS must address the problems described above (the main problem is a lack of interoperability). Meanwhile, the problems of NL-query parsing (morphology, syntax) in general are domain independent and could be solved to some extent uniformly. There are no problems for IT professionals to extract data from traditional relational databases (RDB) due to SQL. The main problem is to automate the translation of a NL-query to a structured SQL-query. To have a unified solution, we should abstract transformation mechanisms from any domain (also, from the domain language and the schema of any concrete RDB).

To solve this problem, we propose using ontology engineering methods. According to Gruber, an ontology is a specification of a conceptualization. In Gruber's work (Gruber, 1993), the process of ontology engineering is determined by the following steps: associating a plurality of "human" terms in the domain with many "machine-readable" classes and/or objects, relations and functions, linking entities and formal axioms that constrain the interpretation of terms and their proper use.

An ontology is based on descriptive logic (Akerkar, 2014) and integrates advances in logical and graphical models of knowledge representation. There are a number of graphical ontology editors that help design and debug ontologies. Standardization of ontology formats (e.g. OWL) simplifies ontology reuse. A knowledge base, represented in ontology format, contains two components – terminological (TBox) and assertion (ABox). A TBox is a finite set of concept inclusions and an ABox is a finite set of assertions.

In our approach, the ontology is built automatically from the legacy RDB's schema according to the rules described in (Dou, LePendu, Kim, & Qi, 2006) and is enriched in an automatic or semi-automatic way with hyponyms, hypernyms, meronym, holonym, and synonyms from a domain ontology, by means of a visual ontology editor, ONTOLIS, developed at Perm State University (Chuprina & Nasraoui, 2016; Chuprina, 2015) or any other visual editor supporting the OWL standard. More than that, external linguistic resources could be used for automatic ontology enrichment, e.g. WordNet (Navigli & Ponzetto, 2012).

An enriched ontology is the basis for the automatic mapping of any NL-query. However automated mapping of elements from a RDB-schema to a conceptual model must address several difficulties. In
particular, names of tables and columns in a RDB do not always exactly represent the semantics of a domain area and might not be used within a casual user’s queries. These names can include cryptic acronyms and numbers, can use multiple languages, or can use non-meaningful names (such as ‘field_1’). More than that, the common convention of naming the relationships by using only the names of the fields connected to the names of foreign keys is not helpful to our approach because real-world names of relations represent the semantics of data with greater power. For example, in a database about the educational process in the university, there could be multiple relations between the table containing information about training programs and the table about teachers: "is an author", "is a reviewer", "use in the teaching process". This leads to the need for additional steps to eliminate the ambiguity of interpretation during the process of mapping the domain concepts to concepts from the extracted ontology. Exacerbating the problem is the fact that the concepts and the relationships in a relational model have the same representation as the relational table.

To simplify the next steps of the transformation of a NL-query to a SQL-query, we demonstrate how the students used one of the existing and freely-available frameworks, Ontology-based Data Access (OBDA). The OBDA approach focuses on providing access to one or more data sources through an ontology mediation – see (Calvanese, De Giacomo, Lembo, Lenzerini, Poggi, et al., 2009) for more details. As a result, the data from the original sources could be reusing SPARQL. One of the main advantages of OBDA is the declarative description of data on the conceptual level and the incremental nature of updates (Calvanese, De Giacomo, Lembo, Lenzerini, & Rosati, 2009). Disadvantages and limitations include:

- Limited applicability due to the casual users’ requirement to formulate queries using SPARQL.
- Difficulties in creating prerequisites (ontologies and mappings).
- Limitations of existing OBDA frameworks.
- Low efficiency of query translation and execution processes.

In the proposed approach, we try to eliminate these disadvantages by means of creating an automatic domain independent NL-query-to-SPARQL-query translator (which is not typical for OBDA-frameworks) and building a bootstrapper that automatically extracts the necessary ontologies and mappings from the RDB-schema. SPARQL to SQL translation is done by OBDA-frameworks automatically if an ontology and correct mappings are provided.

3 Reply Architecture

We have developed an ontology-driven web-service, named Reply, according to the proposed approach, and we use it to demonstrate to the students how to tackle the problems mentioned above. We begin with the demonstration of the Reply web-service architecture and emphasize the advantages of a service-oriented architecture (SOA).

Then we discuss the benefits of using ontology techniques for SOA-systems. These benefits include:

- Logical inference algorithms based on descriptive logic are faster than structured algorithms.
- The SOA system integrates easier with other logical inference algorithms, ontologies, and descriptive languages.
- The SOA system integrates easier with implementations of Semantic Web technologies.

The modularity of the proposed architecture (see Figure 1) simplifies the development process and makes every component reusable. We use a mapping bootstrapper, an ontology editor, and a semantic retrieval module as basic Reply components. The Semantic Retrieval component transforms a NL-query
into a SQL-query for a concrete RDB based on the domain ontology content. An analogous approach could be applied to other structured sources which could be integrated into the RDB using virtualization tools like JBoss Teiid*.

The demonstrated system does a sequence of transformations. A NL-query is transformed to a SPARQL-query by the Reply web-service, developed by the authors. Then the query is transformed automatically into a SQL-query by the OBDA-framework Quest (Ontop, see (Calvanese, De Giacomo, Lembo, Lenzerini, & Rosati, 2009)) based on mapping rules generated by Reply.

NL-query parsing has lexico-morphological, syntax, and semantic analysis sub-steps. To solve the problem of automatic transformation of a NL-query to a SQL-query for legacy RDBs, we developed an approach which automatically discovers concepts in a NL-query and their relations in the domain knowledge base. Freeware stemmers and lemmatizers are used to convert words from an input query into normal form. Problems of homonymy and interpretation ambiguity should be solved using a context-based approach which is out of the scope of this paper. But in general, our solutions are based on natural language independent methods.

A source NL-query, with words in a normal form, is used as an input to a syntax tagging web-service. Syntax pre-processing based on lexico-syntactic patterns (Panchenko, 2013) comes next. The result is then automatically transformed into a SPARQL query, written using terms from the source query, and enriched by hyponyms, hypernyms and synonyms from the domain ontology, to increase the semantic power of the demonstrated approach.

The quality of the knowledge base is crucial to ontology-driven intelligent systems. A visual ontology editor is required for domain ontology design related to the DB-Content Ontology, which is automatically generated by a bootstrapper from the legacy database schema. We use the visual ontology editor, ONTOLIS, for this purpose. The enrichment of the DB-Content Ontology is achieved by the

* [http://teiid.jboss.org/](http://teiid.jboss.org/)
comprehensive means of special services of the ontology editor to integrate by mapping the different ontologies (DB-Content Ontology and the related external domain ontology). The sequence of transformations of the DB-Content Ontology is shown in Figure 2.

Figure 2: Sequence of transformations of the DB-Content Ontology

Several adaptable systems exist for automatic generation of related ontologies based on ontology learning methods, and these were developed at Perm State University. These systems work with Russian and English corpora. Figure 3 shows a fragment of the ontology learning results obtained within the TAISim system, developed by PSU students under the supervision of Prof. Svetlana Chuprina (one co-author of this paper). The paper (Council, 2013) is used as an input source for the ontology learning process, in this example.

Figure 3: The ontology learning process result obtained within TAISim
4 Results and Discussion

We show an example of the result of a NL-query obtained by means of the Reply web-service from the open demo database Adventure Works of personnel working in 'North America' sales territory group as contact persons (see Figure 4).

We distinguish two types of the interface components: configuration components and runtime components. The user interface looks like a typical web search engine interface with the search box and an area for the results. There are settings for the user's individual preferences for the display of search results (preview of images and video, different result view types like list or tables).

The web-interface communicates with the web-services using open standard formats (XML/JSON), supported by most of the popular programming languages. The switchers between different data sources make the proposed system, web single sign on. The system could be easily integrated into an existing web-based information system interface.

During the demonstration of the results of submitting a NL-query to a relational database, we explain the transformation process of the source NL-query. In our example, the input query is «first name, last name and login id of the personnel working in the 'North America' sales territory group as contact persons».

![Figure 4: Reply web-service frontend](image-url)
At first, the query analyzer extracts the classes “Personnel”, “Sales territory” and the object property “Work as contact person”. That object property connects the class “Sales person” with the class “Store”. “Sales person” is a subclass of “Employee”, which is equivalent to “Personnel” for this example. Based on that, the input request is translated into the following SPARQL query of all sales persons who work as contact persons:

```sparql
SELECT DISTINCT ?sp {
    ?sp a :SalesPerson.
    ?store :contactPerson ?sp.
}
```

The words in single quotes are interpreted as a value constraint on a corresponding field, so they are used to restrict the `group`’s value of “Sales territory” individuals corresponding to the selected sales person. The text of the SPARQL query is changed into:

```sparql
SELECT DISTINCT ?sp {
    ?sp a :SalesPerson.
    ?store :contactPerson ?sp.
    ?sp :hasSalesTerritory ?sTerr.
    ?sTerr :SalesTerritory_Group ?sTerrGroup
    FILTER (?terrGroup = "North America")
}
```

Next, the analyzer tries to discover fields to be shown. Class “Sales person” does not have a “Login ID” field, but its super-class does. To get the value, “sales person” is converted into “employee” using conversion rules provided by the knowledge engineer, which are stored as Mapping rules (see Figure 1). After the conversion, the new version of the SPARQL query becomes

```sparql
SELECT DISTINCT ?loginId {
    ?sp a :SalesPerson.
    ?store :contactPerson ?sp.
    ?sp :hasSalesTerritory ?sTerr.
    ?sTerr :SalesTerritory_Group ?sTerrGroup
    FILTER (?terrGroup = "North America")
    ?e a :Personnel.
    ?sp :SalesPerson_BusinessEntityID ?beID.
    ?e :Employee_BusinessEntityID ?beID.
    ?e :Employee_LoginID ?loginId.
}
```

The same conversion applies to first and last names. They are data properties of a super-class and require an explicit conversion. The final version of the generated SPARQL query is as follows:

```sparql
SELECT DISTINCT ?firstName ?lastName ?loginId {
    ?sp a :SalesPerson.
    ?store :contactPerson ?sp.
    ?sp :hasSalesTerritory ?sTerr.
    FILTER (?sTerrGroup = "North America").
    ?e a :Personnel.
    ?sp :SalesPerson_BusinessEntityID ?beID.
    ?e :Employee_BusinessEntityID ?beID.
    ?e :Employee_LoginID ?loginId.
}
```
The final version of the SPARQL query, related to the source NL-query, is used as an input to the OBDA-framework Ontop that automatically generates the following relational database SQL-query:

```
SELECT *
FROM (SELECT DISTINCT
    7 AS "firstNameQuestType", NULL AS "firstNameLang",
    QVIEW5."FirstName" AS "firstName",
    7 AS "lastNameQuestType", NULL AS "lastNameLang",
    QVIEW5."LastName" AS "lastName",
    3 AS "loginIdQuestType", NULL AS "loginIdLang", QVIEW4."LoginID"
AS "loginId"
FROM "Sales"."Store" QVIEW1,
"Sales"."SalesPerson" QVIEW2,
"Sales"."SalesTerritory" QVIEW3,
"HumanResources"."Employee" QVIEW4,
"Person"."Person" QVIEW5
WHERE QVIEW1."BusinessEntityID" IS NOT NULL AND
QVIEW1."SalesPersonID" IS NOT NULL AND
(QVIEW1."SalesPersonID" = QVIEW2."BusinessEntityID") AND
QVIEW2."TerritoryID" IS NOT NULL AND
(QVIEW2."TerritoryID" = QVIEW3."TerritoryID") AND
(QVIEW3."Group" = 'North America') AND
(QVIEW1."SalesPersonID" = QVIEW4."BusinessEntityID") AND
QVIEW4."LoginID" IS NOT NULL AND
(QVIEW1."SalesPersonID" = QVIEW5."BusinessEntityID") AND
QVIEW5."FirstName" IS NOT NULL AND
QVIEW5."LastName" IS NOT NULL
) SUB_QVIEW
```

As the reader (and students!) can see in Figure 4, the result is not only relevant, but also pertinent. Even the result of the intermediate step (SPARQL query) is useful. With a NL interface to the SPARQL generator, which is a component of the Reply web-service, students are implicitly taught how to do this by themselves. Nowadays many Open Data Repositories have a SPARQL-endpoint. This is why the skills of formulating SPARQL-queries are crucial for future knowledge engineers.

The Open Data Portal of Perm Krai is one of the best local Open Data Portals in Russia. Many datasets (statistics and so on) are available for browsing and analyzing. A SPARQL-endpoint is available there for querying the open data (see Figure 5) and we use it in the educational process to teach students how to query and analyze Big Data.
In the future, we believe that our NL to SPARQL translation module could enable a new level of user experience. On the other hand, if more data repositories provide a SPARQL-endpoint, it would be easier for us to integrate them into the system that was described in this paper.

Figure 5: Perm Krai open data portal SPARQL-endpoint

5 Conclusions

We described a web-service that can transform traditional legacy information systems with relational databases into intelligent or smart information systems with a NL interface. The main mechanism to do this transformation is turning a natural language query into a SQL-query for structured data sources that can help data scientists and data engineers, as well as any IT-specialists and casual users in the decision making process. The main paradigm of the proposed approach is Ontology-Based Data Access. According to the proposed approach, the transformation of a legacy IS into an intelligent IS does not need any modification of the original information system source code. The paper describes the role of a NL-query transformation module, the basic concepts of its implementation and its integration method with existing OBDA-frameworks. We used the step-by-step demonstration of the proposed approach and its implementation during the educational process in Perm State University to help bridge the talent gap in data analytics and to transform a traditional IS into a more intelligent IS with a NL interface, which is helpful in the context solving Big Data problems.
Acknowledgements

The reported study was partially supported by the Government of Perm Krai, research project No.C-26/004.08 and by the Foundation of Assistance for Small Innovative Enterprises, Russia.

References


