A Natural Language Interface for Data Warehouse Question Answering

Nicolas Kuchmann-Beauger[1,2] and Marie-Aude Aufaure[2]

[1] SAP France, 157/159 rue Anatole France, 92309 Levallois-Perret Cedex, France,
nicolas.kuchmann-beauger@ecp.fr
[2] Ecole Centrale Paris, MAS laboratory, 92290 Chatenay-Malabry, France,
marie-aude.aufaure@ecp.fr

Abstract. Business Intelligence (BI) aims at providing methods and tools that lead to quick decisions from trusted data. Such advanced tools require some technical knowledge on how to formulate the queries. We propose a natural language (NL) interface for a Data Warehouse based Question Answering system. This system allows users to query with questions expressed in natural language. The proposed system is fully automated, resulting low Total Cost of Ownership. We aim at demonstrating the importance of identifying already existing semantics and using Text Mining techniques on the Web to move toward the users's need.

Keywords: Question Answering, Natural Language Interface, Data Warehouses

1 Introduction

Business Intelligence (BI) aims at providing a better understanding of the business environment to make decision. In this perspective, software companies have developed applications and tools that ease this understanding through data warehousing and data visualization which leads to better decision-making.

To access these data, all databases (DB) have an interface from which any user may write a query. This query is usually a technical query expressed in a specific language (e.g. MDX or SQL).

An issue is that such language is not *natural* to a non-expert user. In order to address this, several tools have been proposed such as SAP BusinessObjects Explorer [9]. One notice that once a user has entered a query, they modify it through classical operation (roll up, drill down). Indeed, building the best user interface for data warehouses is a subject that raises lots of interests.

The question on how to best display an answer for a real user's question belongs to the Information Retrieval (IR) domain. Systems that handle questions expressed in Natural Language (NL) are called Question Answering (Q&A) systems. Traditional IR systems allow users to express their queries using keywords or using a restricted syntax.

The basic approach in IR consists in identifying keywords or known entities in the user's need expression, and in linking these entities to objects defined in

the data model. We have implemented this approach, and we propose an original method to get better results in this context. A thesaurus is used to rewrite the user query when no answer has been found.

In the following, *question* is used for a well-formed question expressed in NL, while *query* stands for a query that the system can render (e.g. expressed in SQL). The expression *user's query* stands for *question*, or a user question that is not a well-formed question but composed of keywords (e.g. "Sales revenue France 2008" is a *user's query* that refers to an abstract *question*, one of whose occurrence is "What is the sales revenue in France in 2008?").

Our experiments show that the keyword-based aprroach is not sufficient, and that the same query may be expressed with different words. We evaluated the relevance of the keyword-based approach from a set of hundreds of complex business questions (see an extract table 1). The toy dataset to which the questions

Table 1. Extract of questions used in the evaluation of the "String Matching" component

domain	**question**
eFashion	Which store has not sold which product category?
eFashion	How many orders have been generated in New York?
eFashion	Where customers have not bought which product?
eFashion	What was the most profitable product category?
eFashion	What product quantity was ordered in Chicago by quarter and what was the average for all quarters?
eFashion	What revenue did we achieve in Florida?

belongs is related to business sales and orders ("eFashion" dataset). The questions could all be answered by the system. Among these questions, only 52% contain at least one object recognized in the component (which does not mean that the analysis of the component is correct).

2 Related Work

Q&A is one of the first applications of Artificial Intelligence, and the first proposals were based on structured data [7, 15].

These systems deal with databases interfaces, and more specifically natural language database interfaces [2]. [11] is an example of natural language interface to a XML database. [5] proposes such a system that links keywords present in the user's question into a SQL-statement. The data are split into several tables that have to be joined to get the requested information. Usually, a grammar dedicated to NL interpretation is needed [8] which is a limitation, because the NL processing is tailored to the database itself.

[13] aims at determining whether a question can be answered by the system, and if so it asserts that the answer corresponds to the correct interpretation. In

the field of enterprise Q&A, one generally prefer retrieving pieces of answers, even if they are not the exact interpretation. [10] is a system that can be used with any database, however it requires resources in addition to the database. Recently, [6] proposed a Q&A system dedicated to enterprise questions but also to general domain questions.

Building an ontology [14, 3, 16] is often used to support Q&A, especially in the semantic Web community. Ontologies are often used in the Question Rewriting component [4], [1]. The ontology is then used to translate the user's queries into a technical query. Accessing a database through an ontology in not new: e.g. [12] presents how an ontology can be used in order to get data from a relational database.

We leverage traditional Q&A techniques with the specificities of an enterprise IR system. In this perspective, a user-specific ontology is being automatically built and enriched, this will be further discussed below.

3 Architecture

In this paper we are interested in the first component, aiming at describing the semantic analysis of the user's query, as described in the figure 1. The Named

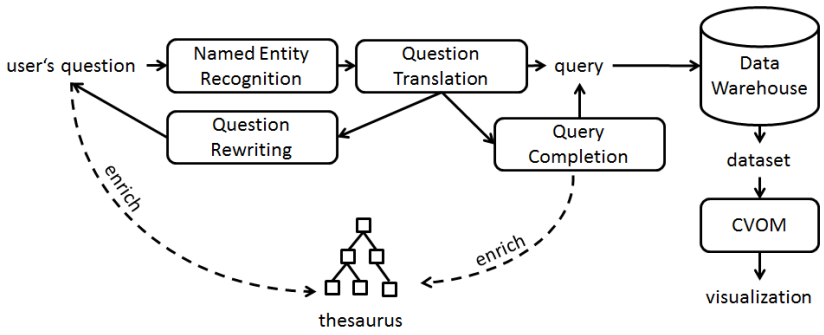

Fig. 1. Analysis of the user's question

Entity Reconizer (NER) aims at identiying *known objects* of the data model, that comprise dimensions, measures and values of dimensions. The Question Translation component build a technical query from a user's question. The Question Rewriting component rewrites the query if no answer can be retrieved. The Query Completion component is used when queries are not expressive enough (e.g. when no dimension is comprised in the query). The CVOM[3] component provides a visualization to a dataset. In this paper, we focus on three components: the Question Tranlation component, the Question Rewriting component and the thesaurus-based Q&A.

[3] SAP BusinessObjects Common Visual Object Modeler

4 Question Translation

Identified semantic units from the user's question are translated into a machine-readable query (technical query), which is a graph composed of objects and constraints described in the data warehouse model. For example, the user's question "What are the sales revenue in France in 2008" will be translated into the following technical query:

```
AGGREGATE MEASURE 'Sales Revenue' AGAINST guessed DIMENSION
FILTERING ON 'France' BEING 'Country'  AND '2008' BEING 'Year'
```

There is here a direct link between the entities expressed in the user's query and the objects modeled in the data warehouse. If the query is empty (no object can be recognized) the Question Rewriting component provides new question formulations from which the Question Translation component is called. If the query is not empty but does not return any answer, the Query Completion component is called. We exploit scores from a search service[4] to infer semantic closeness between terms or expressions from the user's query and labels. Scores are the number of pages that contains the expression.

Let consider the question "What are the admission fees in Asia?". No entities are identified in this question. One of the rewritten questions is "What are the admissions in Asia?" (one statement in the thesaurus is that *admission fees* and *admissions* are synonyms, and Admissions is an object defined in the data model. In this example, "Asia" is not defined in the data model, and no row in the data contains this token. The following section shows how it is possible to deal with such cases.

5 Thesaurus

Our current work implements automatically built thesaurus from structured data. Thesaurus building is done in three-steps: 1) building of a minimal thesaurus, 2) validating this minimal thesaurus and 3) adding terms to be validated in the thesaurus.

5.1 Minimal thesaurus

The minimal thesaurus is built from objects defined in the data model. They are represented in the thesaurus by terms as well as values of dimensions that are represented as terms sharing the *is-a* semantic relationship with the concerned dimensions. For each terms, WordNet is used to add relationships (e.g. synonymy relationship) and new terms (e.g. synonyms). The sense disambiguation is the main focus of the thesaurus validation. Our heuristic consists in choosing the first sense, that corresponds to the most common sense.

[4] See Yahoo BOSS at http://developer.yahoo.com/search/boss/.

The validation of the thesaurus is performed as the user queries the system. The algorithm is based on the identification of terms in the user's query which are semantically linked with other terms in the thesaurus.

The figure 2 presents terms of the minimal thesaurus automatically built from the *eFashion* dataset. The top concept is labelled "TOP". The first-level edges

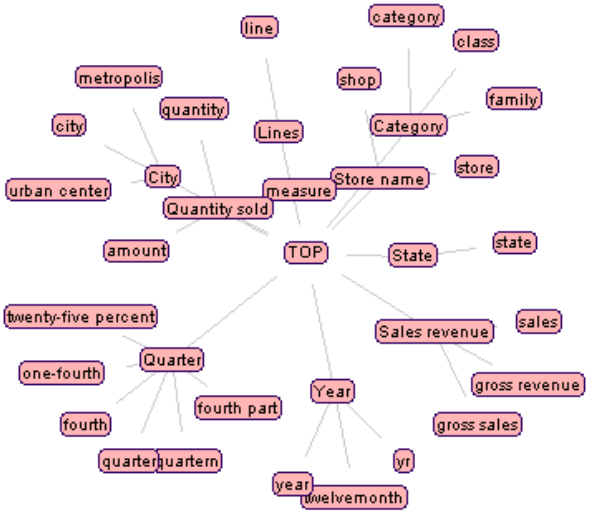

Fig. 2. Minimal thesaurus

are *is-a* relationships, while other relationships are synonymy relationships.

5.2 Learning approach

To address the problem of unknown terms (such as "Asia" in the user's question "What are the admission fees in Asia?"), we try to categorize them using known terms as labels. Our problem is similar to classification problems. The algorithm 1 presents how new terms are added to the minimal thesaurus. In our example, if we remove the known entity (*admission fees*) and the stop words (defined in a specific dictionary), the remaining is "Asia". The algorithm returned two related entities, "Malaysia" and "Japan", both values of the dimension *Admission Country*. The algorithm ranks these entities according to the frequency of the related dimensions, and adds them to the query as filters.

New instances are also added when the user queries the system. We use patterns to get new instances from existing ones, similar to those defined in the Ontology Design Pattern (ODP) portal[5].

[5] Go to http://ontologydesignpatterns.org/wiki/Main_Page for more details.

Algorithm 1 Query completion

Require: question$\in L\star$
```
question←remove_known_entities(question)
question←remove_stop_words(quesiton)
for word in question.words do
    abstracts←web_search(word)
    for abstract in abstracts do
        for token in abstract.words do
            if is_known(token) then
                add_related_entity(get_entity(token))
            end if
        end for
    end for
end for
rank(related_entities)
```

6 Query Reformulation

Query Reformulation aims at rewriting queries so that the user's need is better satisfied. Two common ways of rewriting a query is expanding or refining it. Other kinds of reformulation replace terms by other ones sharing a specific relationship. This process is iterated, until a valid technical query is available.

In our present work, we only use the synonymy relationship amoung terms to reformulate user's queries. Word-level N-grams are identified in the user's query, and matched to the thesaurus. The corresponding terms are listed, and semantically related terms are retrieved trom the thesaurus using the SPARQL[6] query language. An ordred list of rewritten queries are then processed by the system, and corresponding answers are presented to the user. The order is defined by the semantic closeness of retrieved terms.

7 Evaluation

We consider a movie-related dataset (containing data about movies with associated business data such as the corresponding admission fees and budget). One typical query would be "What are the admission fees in *this* country". Let analyze the user's query "Compare the admission fees in Europe and in America". The traditional keyword-based method cannot retrieve any information for two reasons: 1) the modeled fact is not called "admission fees", 2) "Europe" and "America" are not defined. The first concern is resolved by the reformulating component using the minimal ontology. The second concern need a deeper analysis because the terms "Europe" and "America" are not found in the thesaurus. Some of the patterns used to validate "Europe" from the user's terms is "Europe is composed of $\star$" and "$\star$ compose Europe". Table 2 shows the results of the example. The entity type `COUNTRY` gets the best score, which means that the terms

[6] See http://www.w3.org/TR/rdf-sparql-query/ for more details.

Table 2. Entities identified in a Web corpus dedicated to ontology enrichment

Entity type	found entities	distinct found entities
PLACE_OTHER	3	3
TIME_PERIOD	3	2
ORGANIZATION	19	9
PLACE_REGION	117	17
COMPANY	7	6
COUNTRY	106	38
PERSON	13	12
DATE	9	9
CITY	4	4

must be of the COUNTRY type. Once the new terms are discovered, the algorithm selects known terms that best represent the user's need. In this example, the retrieved terms of type COUNTRY are: United States, Belarus, Portugal, Slovakia, Greece, Spain, Ireland, England, Northern Ireland, Luxembourg, Switzerland, Italy, Great Britain, France, Moldova, Liechtenstein, Denmark, Netherlands, the Netherlands, Britain, US, Finland, UK, Austria, Wales, United Kingdom, Andorra, Scotland, America, Hungary, Czech Republic, Slovak Republic, Belgium, Poland, Romania, Bulgaria, Russia, Norway, Germany, Sweden. One notice that this output is not perfect ("United States"). In our example, Norway, Germany and Sweden were already known, so the rewritten query is composed of these three filters.

8 Conclusion and Future Work

We have proposed a Q&A system based on structured data enabling users formulating their queries in NL. The system is composed of a string matching component performing keyword-based Q&A and of a learning thesaurus. This thesaurus is automatically built from the data model, and is enriched through users' queries. It is used to rewrite the queries.

One improvement will be turning the thesaurus into an ontology, wich will provide a reasoning feature. The ontology could be used for other purpose than question rewriting, like gathering data from unstructured sources to validate provided answers. Improvements when formulating search patterns (e.g. using lemmas) are also planned.

References

1. An, Y., Borgida, A., Mylopoulos, J.: Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In: Meersman, R., Tari, Z., Hacid, M.S., Mylopoulos, J., Pernici, B., Babaoglu, Ö., Jacobsen, H.A., Loyall, J.P., Kifer, M., Spaccapietra, S. (eds.) OTM Conferences (2). Lecture Notes in Computer Science, vol. 3761, pp. 1152–1169. Springer (2005)

2. Androutsopoulos, L.: Natural language interfaces to databases - an introduction. Journal of Natural Language Engineering 1, 29–81 (1995)
3. Ben Mustapha, N., Zghal, H.B., Aufaure, M.A., ben Ghezala, H.: Semantic search using modular ontology learning and case-based reasoning. In: EDBT '10: Proceedings of the 2010 EDBT/ICDT Workshops. pp. 1–12. ACM, New York, NY, USA (2010)
4. Bizer, C.: D2r map - a database to rdf mapping language (2003)
5. Chaudhuri, S., Das, G.: Keyword querying and ranking in databases. Proc. VLDB Endow. 2(2), 1658–1659 (2009)
6. Ferrucci, D.A., Brown, E.W., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J.W., Nyberg, E., Prager, J.M., Schlaefer, N., Welty, C.A.: Building watson: An overview of the deepqa project. AI Magazine 31(3), 59–79 (2010)
7. Green, B., Wolf, A., Chomsky, C., Laughery, K.: Baseball: an automatic question answerer pp. 545–549 (1986)
8. Hendrix, G.G., Sacerdoti, E.D., Sagalowicz, D., Slocum, J.: Developing a natural language interface to complex data. ACM Trans. Database Syst. 3(2), 105–147 (1978)
9. Hilgefort, I.: Inside SAP BusinessObjects Explorer. SAP Press, Bonn, Germany (2010)
10. Knowles, S., Mitrovic, S.T.: A natural language database interface for sql-tutor (1999)
11. Li, Y., Yang, H., Jagadish, H.V.: Nalix: an interactive natural language interface for querying xml. In: SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. pp. 900–902. ACM, New York, NY, USA (2005)
12. Poggi, A., Lembo, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Semantics 10, 133–173 (2008)
13. Popescu, A.M., Armanasu, A., Etzioni, O., Ko, D., Yates, A.: Modern natural language interfaces to databases: composing statistical parsing with semantic tractability. In: Proceedings of the 20th international conference on Computational Linguistics. COLING '04, Association for Computational Linguistics, Stroudsburg, PA, USA (2004), http://dx.doi.org/10.3115/1220355.1220376
14. Serhatli, M., Alpaslan, F.N.: An ontology based question answering system on software test document domain. World Academy of Science, Engineering and Technology (2009)
15. Woods, W.A.: Progress in natural language understanding: an application to lunar geology. In: AFIPS '73: Proceedings of the June 4-8, 1973, national computer conference and exposition. pp. 441–450. ACM, New York, NY, USA (1973)
16. Zghal, H.B., Aufaure, M.A., Mustapha, N.B.: A model-driven approach of ontological components for on-line semantic web information retrieval. J. Web Eng. 6(4), 309–336 (2007)