Natural Language Interfaces for DataWarehouses

Nicolas Kuchmann-Beauger, Marie-Aude Aufaure

Natural Language Interfaces for Data Warehouses

Nicolas Kuchmann-Beauger* and Marie-Aude Aufaure**

*SAP Research – 157/157 rue Anatole France – 92309 Levallois-Perret Cedex
nicolas.kuchmann-beauger@sap.com,
**École Centrale Paris – Grande Voie des Vignes – 92290 Châtenay-Malabry
marie-aude.aufaure@ecp.fr

Abstract. Data warehouses process a huge mass of data, that are often modeled in a way humans hardly understand. In order to make the search paradigm more accessible to end users, some efforts have been made in the field of Business Intelligence. However, expressing information need as a structured query is still an artificial task. This explains why "natural" approaches are preferred nowadays. In this paper, we present a Question Answering (Q&A) system for structured data in a context of BI. Some benefits of our proposal are described through an iPhone/iPad application and through an HTML prototype.

1 Introduction

In recent years, a new class of search systems, such as Wolfram|Alpha [1], became popular: question answering systems that operate on well curated knowledge bases and thus deliver highly reliable answers to user questions. Industrial solutions have been designed by the BI community to support users in formulating queries. However, the query models are often oversimplistic compared to the complex questions that users have in mind. Query interfaces in commercial BI tools allow users to combine dimensions and measures to build queries.

New device capabilities (increased smartphones and tablets memory and storage) as well as the generalization of Internet access to mobile devices bring new challenges. In particular, recent success of speech-to-text technologies like Siri [2] has made natural search interfaces popular. This observation, and the fact that users are more comfortable to use such query interfaces compared to very structured ones have been pointed out by Hearst (2011). As a result, there is a strong requirement from non-expert employees (e.g., sales representatives) to access their data on their mobile devices and to formulate queries without any knowledge on the underlying query language. In addition, mobile devices bring also new information about users (e.g., geographic location). Such information are processed by context-aware algorithms.

As a proposition to overcome the shortcomings of existing BI tools, and taking into account new trends in terms of context sharing and mobility in information retrieval, we present in this paper an answering system for data in warehouses. The remaining of the paper is structured as follows: Section 2 presents hierarchies in multidimensional analysis, and Section 3 introduces

1. designed by Mathematica, see `http://www.wolfram.com/mathematica/`.
2. For more information about Siri: `http://www.apple.com/iphone/features/#siri`

the architecture of the proposal. We present the question parsing component in section 4. We detail the semantic mapping of NL questions to structured queries in section 5. Then, we survey the field of natural language interfaces in section 6. Preliminary experiments that have been made are presented in section 7.

2 Hierarchies in multidimensional analysis

Most multidimensional analysis define *hierarchies* which is the basic information to do classical OLAP operations (e.g., *drill-up* and *drill-down*).

2.1 Hierarchies

Hierarchies in multidimensional analysis are modeled as an ordred set of dimensions where the set corresponds to the transitive closure for a given hierarchical relation. For example, the set of dimensions: $\{city, country, year, quarter, region, state, store\}$ can be divided in $h_1 = (city, state, country, region)$, $h_2 = (quarter, year)$ and $h_3 = (store)$. Those three lists correspond to different hierarchies.

2.2 OLAP operations

Modeling hierarchies in multidimensional analysis result in interesting behavior when triggering OLAP operations. We focus in particular on drill-down and drill-up below.

Drill-down. Drill-down consists in focusing on a value of a dimension, and in providing more detailed information about the fact that corresponds to this dimension value. In practice, users manipulate charts (rather than datasets). The drill-down operation is triggered when user selects a value of a dimension in the chart itself (e.g., user double-clicks on a chart item, for instance a bar, that corresponds to a dimension value). In the case when this dimension is related to other dimensions in the same hierarchy, the selected dimension is used as a filter for computing the facts, and the displayed dimension is the next available dimension in the hierarchy. For instance, if the user is exploring a bar chart which displays the sales revenue per year (each bar corresponds to a year) and if she selects a bar, a new chart is being created, where all facts correspond to the chosen dimension value. The next dimension in the time hierarchy is displayed (e.g., quarter); as a result the chart will display the sales revenue per quarter for a given year. If the chosen dimension is not part of a hierarchy (or is at the lowest level in the hierarchy), the operation will choose an other compatible dimension basing on some heuristics (see section 4.4 for the notion of compatibility).

Drill-up. Drill-up (or roll-up) is the inverse operation. In the case of a query where the main dimension (usually the one displayed on the x-axis on a chart) is not at the top level in the hierarchy, the new dimension used for computing the facts would the next available dimension. This recomputation bases on the aggregation function, defined for measures in the data model (sum, count, maximum, ...).

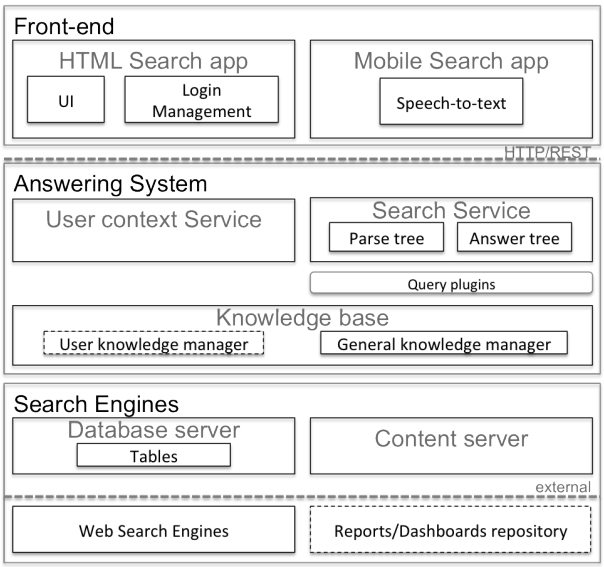

FIG. 1 – *Architecture of the answering system*

3 Architecture overview

The system is composed of three main components: 1) the front-end for different kinds of devices; 2) the answering system itself; and 3) different search engines. Figure 1 presents the overall architecture of the proposal. Dashed boxes correspond to components under development.

3.1 Data

The system architecture is general designed to operate as federated search engine to support also traditional search systems. In addition, it provides a framework for question answering that operates on the various data structures. The database server (upper-left corner in the third top-down component in figure 1) gives access to the data warehouse modeled as presented in the introduction. The content server (upper-right corner in the third component) is a storage system which may contain various kinds of data, like employee's documents, or more structured data like the social network of employees (the data stored in this content server are not used so far for answering users' questions). In the lower-part of the third component, external sources like Web search engines (Yahoo!, Bing, ...), freely available structured data (Freebase, Wolfram|Alpha) or empoyees' reports and dashboards are provided as answers to questions expressed in natural language. The system presented in this paper has been experimented so far with the database server.

3.2 Answers

Answers are of different types: structured answers (resulting of the execution of a structured database query), semi-structured answers (reports or dashboard coming from the external repository) or unstructured (Web pages retrieved by search engines).

In this paper, we focus on structured answers (i.e. database queries whose execution lead to a dataset and the generation of a chart). For results, we generate a title which captures the system-understanding of the question. We use custom patterns to generate such titles from the internal graph representation of the query (see section **??**). Besides, structured results that are displayed as charts are linked with external applications dedicated to explore BI charts (those applications permit for instance to easily and dynamically change the query being displayed within a dashboard).

3.3 Main components

Our system is composed of following components:

- **Question parsing** The question parsing component aims at analysing the input question. Its output is a tree data structure where nodes correspond to *annotations*.
- **Semantic mapping**ăThe parse tree is associated to an internal logical representation of the query. This mapping is performed using custom linguistic patterns.
- **Query generation** The internal query representation is translated into the target query, which is then executed. The resulting dataset is then used to generate automatically charts and reports.

We detail the two first components in sections 4 and 5. The last one is not further detailed here because of lack of space.

4 Question parsing

Natural language questions are subject to several processing steps to prepare the data structures that can be used to formulate a technical query that can be executed on the data warehouse. The target structure has the form of a tree, where the root node represents the query and the leave nodes potential semantics of parts of the query. We distinguish two types of semantics: recognized entities (e.g., measures and dimensions), which we call *annotations* and natural language phrases that define further constraints of potential queries (e.g., a ordering and limit expression such as 'best 5'). We will refer to the latter as *natural language features* in the following. Unlike traditional interfaces, edges do not reproduce syntactic information but correspond to annotation types. Figure 2 is an example of parse tree generated for the question "Revenue per country in 2009". The question parser is composed of several components: tokenizer, stemmer, POS-tagger, entity tagger and advanced pattern-matching mechanism. We focus on the two latter in sections 4.1 and 4.2. In addition our system can rely on domain specific background knowledge such as the user context and usage monitoring (see sections 4.4 and 4.3).

FIG. 2 – *Example of parse tree generated for the question "Revenue per country in 2009"*

4.1 Entity recognition

The term named entity recognition (NER) is commonly referring identification of rigid designator in text. In our case, the entities of interest are defined in the data warehouse's data and meta-data. Thus, we rely on a purely dictionary based approach that can be extended with synonym lists, variant generators (e.g., to generate token-n-grams) and scoring algorithms. In particular, our entity recognition component bases on commercial software (SAP BusinessObjects Text Analysis) that automatically loads data and meta-data from the data warehouse, generates token-n-grams of variable length (e.g., for 'Sales Revenue': 'Sales' and 'Revenue'), performs a TF-IDF-like scoring of the variants and removes variants with a score lower than a certain configured threshold (e.g., we keep only 'Revenue' and 'Sales Revenue' in our dictionary).

4.2 Natural Language Features

In addition to the recognition of entities, we have defined a set of pattern-matching rules that describe BI-specific linguistic information (i.e. natural language features) from questions (e.g., an order by). A rule may consist of ordered or unordered alternations of tokens, stems part-of-speech tags or regular expressions. In addition they can include output delimiters, e.g, to express that we want to use from 'best [0-9]' the number in our final technical query. In addition there is an API to define post-processing steps for output delimiters. This is useful to compute for instance query parameters for question that contain "in the last 5 years". The output delimiter would in this case export '5' which is used to compute '2012', '2011', etc.

4.3 Contextual expressions.

The system presented in this paper has been implemented within a contextual framework (not of the scope of this paper). However, they can be used together with the post-processing mechanisms described above to formulate technical queries. The framework is illustrated in the architecture figure 1 as "User context Service" and "User knowledge manager". It enables context-aware question processing, when the user (logged on in the corporate network) interacts with various applications, and her actions are monitored. We make the distinction between following contextual information integrated in the question parsing stage:

- Geographic context: geographic expressions in a users' question like "there" or "near my position" make reference to the contextual geographic information about the user. This is particularly interesting when user is on a mobile device.
- Social network context: In our case the user is part of a 'social corporate network'. It consists of hierarchical relations (like "X is manager of Y") and possibly of information related to teams (like "X works with Y on the project P"). Corresponding expressions are for instance pronouns. In the question "How is the revenue of my stores?" , the user is asking about facts on the stores she might be involved in (for example stores that she manages).

4.4 Domain-specific knowledge

In addition, our system leverages further information, which helps to formulate technical queries:

Data model. The domain-dependent konwledge base is a lexicon that models the terminology used by experts of the domain. This lexicon is created when users desgin the data models of the data warehouse (see section 2.1, and extended using tools like WordNet (synonyms).

Compatibility. In addition to terminology, the knowledge base also defines compatibility between concepts in the domain. Two concepts are said *compatible* if they can be used together to produce a valid database query. This information is made available at the database schema level. This is particularly important when the system tries to disambiguate a query (for instance, a query where no measure can be found, like "Best performing stores"). One expects in such cases that only compatible measures or dimensions are suggested to produce a valid query.

Usage statistics. An additional domain-specific knowledge is composed of *usage statistics*, which are statistics about how measures and dimensions are used together (reports and dashboards). Such reports may be produced by the user, or a group of similar users. For security reasons, we have considered so far only statistics about the user herself (and not other users that might have a similar profile). We use such usage data for query auto-completion (the system suggests how to complete the query that the user is typing in the search box). In this scenario, we use a n-gram like model, where the rank of a dimension or a measure is computed on the basis of the dimensions and measures that have been already typed by the user.

5 Semantic mapping

The semantic mapping consists in associating a parse tree to semantic information, *i.e.* information on how words and/or phrases of the question are being translated into fragments of the target database query, and eventually how to combine those fragments to produce the final database query. In classical NL interfaces, the parse tree also provide information on the syntactic structure of the question (see section 4), but this is not our case. Our proposal consists in using a generic pattern-matching approach, that creates an internal abstract query representation out of the parse tree of the NL question. The pattern-matching approach that we have implemented is introduced in section 5.1. To model the abstract query representation, we have proposed a conceptual model (see section **??**). The second step of the mapping consists in translating this internal representation into the target database query.

5.1 Linguistic patterns

We adopt a pattern-matching approach to associate annotations in the NL questions to the query representation. The parse tree is encoded in the RDF formalism, and the patterns are written in SPARQL. We have chosen this formalism because it is a natural way of representing different kinds of knowledge, and above all because it originally supports pattern-matching mechanisms. Another good reason for choosing RDF in this case, is because writing constraints is natural. Pieces of information presented in section 4 are represented as *triples* in the sense of RDF (see for example the graph created for the question "Revenue per country in 2009" figure 2). The SPARQL pattern builds a RDF graph (see below) which corresponds to the internal query representation.

6 Related Work

NL interfaces for structured data is an early topic of AI. Early systems like BASEBALL by Green et al. (1961) hardcode the way question words should be associated with database elements. In those systems, users' questions were parsed syntactically, and semantic information was added to the parse tree using a dictionary Those systems are not easily portable to other application domains, and have a limited linguistic coverage.

Reasearchers focused then on linguistic coverage. Their goal was to increase the ability of the system to understand complex questions expressed in NL. This was permetted thanks to advances of NL processing tasks. The most significant system might be CHAT-80 by Warren and Pereira (1982) which is the ancestor of many systems in that period. This system, entirely implemented in Prolog, answers questions in NL for a Prolog database. The NL understanding of the system also comprises English determiners and quantified expressions. The main shortcomings of those systems is the domain portability. Indeed, they required a significant customizing effort that would allow to port them to other application domains. As an answer to this issue, a wide range of systems use learning algorithms to learn domain-specific knowledge. To the best of our knowledge, Miller et al. (1996) are the first to propose a learning algorithm associated with a fully statistical model to translate NL questions to structured queries for databases. In this proposal the syntactic/semantic parse tree is being recursively constructed

basing on a statistical model. The system proposed by Zettlemoyer and Collins (2005) combines both syntactic and semantic information in the question parsing component. The original approach consists in learning the grammar as well as the dictionary used by the parser.

Recently, researchers have renewed their interest for NL interfaces for structured data with the emergence and the success of the semantic web, and the need for user-friendly interfaces for such heterogeneous data. Han et al. (2010) propose a system where no domain-specific knowledge is needed any more. Translation rules are being discovered in the data themselves. Also, the interesting and innovative idea is that the linguistic coverage exactly corresponds to the semantic coverage, *i.e.* the questions that can be answered in the database. To perform this, authors generate structured query templates from all possible database queries. Besides, templates of NL questions are associated with such query templates. When users query the system, the question is compared with the set of question templates the system knows. In the case where no question template can be retrieved, users' feedback is involved to increase the system's performance using user-friendly authoring tool that provides domain-specific konwledge about the question and target query templates.

7 Experiment

We have experimented the system described in this paper with two front-ends (see figure 1): an HTML application available on a desktop as well as on mobile devices, and a native application for iPhone/iPad. The platform is still under development to extend it beyond the features presented in this paper. We present both sides of our experiment below.

7.1 HTML application

The HTML application is written in HTML/Javascript. The communication between the client and the server is done through the HTTP protocol (REST architectural style). The welcome page is simply composed of the search box and a login button. Login enables context-aware capabilities described in section 4.2. Figure 3 is an example of the answer provided for the query "revenue per city". On the result page, the user can swith facets ("corporate" corresponds to the trusted data coming from the corporate data warehouses while "Web" corresponds to unstructured documents retrieved on the Web, and that are not of the scope of this paper). Charts that correspond to answers are entitled to describe the answer to the user.

7.2 Mobile native application

The native application offers similar features as the HTML one but is better integrated in the mobile environment and appear as a standalone application. We added a speech-to-text component in this experiment in order to overcome shortcomings of such devices like the usability of the screen-integrated keyboard. This feature allows users to pronounce their queries instead of typing. The current prototype works pretty well for native speakers in English and German, but not no good for other accents. An improvement would be to have a library that also supports non-standard accents.

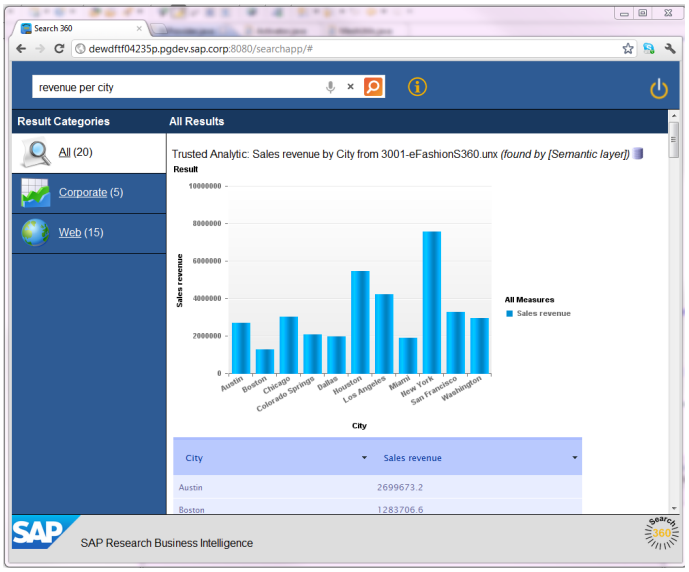

FIG. 3 – *Example of answers for the query "revenue per city"*

8 Conclusion

We have proposed an interface to data warehouses that permits users to formulate queries in NL as well as using keywords. Support of both ways of querying the system are needed, because keyword is the most popular querying style on the Web, while NL draws more and more researchers' attention, because of the new capabilities of mobile devices, and because of the recent advances in the field of voice recognition. One of the benefits of our proposal lies in the fact that there is no strong syntactic requirement in the way users are expected to query the system (unlike similar approaches). We have also described some aspects of the contextual framework that has been integrated to the system. In particular, this framework permits to integrate other kinds of information available accross the corporate network and coming from context-aware applications, which makes our system scalable. Moreover, results that we propose are integrated with other classic BI tools to explore data.

The system aims at generating automatically charts and reports in cases of more complex queries for BI purposes. This specific context has also driven the design of the internal query representation. The overall system is however portable to other domains, the minimum resource to provide is the data model which defines the concepts used in the database.

We have proposed an experimentation through two interfaces: an HTML client and an iPhone/iPad application in order to validate our choices. As a work-in-progress there is still a real evaluation to make. The system also needs to be optimized, so that it answers queries in less than one second. There is a range of improvements that we plan to investigate, for instance integrating a syntactic parser as one of the natural language features presented in section 4.

References

Green, Jr., B. F., A. K. Wolf, C. Chomsky, and K. Laughery (1961). Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, IRE-AIEE-ACM '61 (Western), New York, NY, USA, pp. 219–224. ACM.

Han, Y.-J., T.-G. Noh, S.-B. Park, S. Y. Park, and S.-J. Lee (2010). A natural language interface of thorough coverage by concordance with knowledge bases. In *Proceedings of the 15th international conference on Intelligent user interfaces*, IUI '10, New York, NY, USA, pp. 325–328. ACM.

Hearst, M. A. (2011). 'natural' search user interfaces. *Commun. ACM 54*(11), 60–67.

Kaufmann, E. and A. Bernstein (2007). How useful are natural language interfaces to the semantic web for casual end-users? In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, Berlin, Heidelberg, pp. 281–294. Springer-Verlag.

Miller, S., D. Stallard, R. Bobrow, and R. Schwartz (1996). A fully statistical approach to natural language interfaces. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, ACL '96, Stroudsburg, PA, USA, pp. 55–61. Association for Computational Linguistics.

Thollot, R., N. Kuchmann-Beauger, and M.-A. Aufaure (2012). Semantics and usage statistics for multi-dimensional query expansion. In *Database Systems for Advanced Applications*, Lecture Notes in Computer Science. Springer. To be published.

Warren, D. H. D. and F. C. N. Pereira (1982). An efficient easily adaptable system for interpreting natural language queries. *Comput. Linguist. 8*, 110–122.

Woods, W. A. (1973). Progress in natural language understanding: an application to lunar geology. In *Proceedings of the June 4-8, 1973, national computer conference and exposition*, AFIPS '73, New York, NY, USA, pp. 441–450. ACM.

Zettlemoyer, L. S. and M. Collins (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, pp. 658–666. AUAI Press.

Résumé

Les entrepôts de données manipulent de très gros volumes de données et les modélisent de façon complexe. Pour rendre la tâche de recherche d'information plus accessible à l'utilisateur final, de gros efforts ont été fait dans le domaine de l'informatique décisionnelle. Pourtant l'expression d'un besoin d'information sous la forme d'une requête structurée reste artificielle d'où l'attrait pour les interfaces dites naturelles. Nous présentons un système de questions/réponses pour données structurées dans un contexte BI. L'utilité de notre approche est décrite à travers une expérimentation sur iPhone/iPad ainsi que sur un client de type HTML.