Natural Language Front-End for a Database

Boris Galitsky
Birkbeck College University of London, UK

INTRODUCTION

Whatever knowledge a database contains, one of the essential questions in its design and usability is how its users will interact with it. If these users are human agents, the most ordinary way to query a database would be in the natural language (Gazdar, 1999; Popescu, Etzioni, & Kautz, 2003; Sabourin, 1994). Natural language question answering (NL Q/A), wherein questions are posed in a plain language, may be considered the most universal but not always the best (i.e., fastest) way to provide the information access to a database. One should be aware that approaches to data access, such as visualization, menus and multiple choice, FAQ lists, and so forth, have been successfully employed long before the NL Q/A systems came into play. In the following, I discuss situations in which a particular information access approach is optimal. The five basic means to access (i.e., search) the data with respect to the search methodology is highlighted:

1. Looking through the data itself
2. Consulting the explicit enumeration of choices (e.g., menus, lists, combo boxes)
3. Structured language-based database querying
4. Keyword search
5. NL Q/A (as a front-end to a database or to unstructured data)

For many data access problems, 1 and 2 compete with 3 through 5 in terms of efficiency. Frequently, explicit browsing of the data or using intermediate steps is sufficiently convenient; however, 1 is good for a limited amount of data, 2 requires data structuring and is not flexible, and 3 is used for fully structured knowledge with relational links. Note that database querying may involve the other (i.e., non NL) means: 1, 2, and 4 to build a query that runs against the data source. The most powerful approach to data management seems to be 3, wherein the retrieval is easily and naturally combined with the update. Also, it does not require data modification to provide the NL search, thereby transitioning to 5.

Nowadays, in a majority of applications, the number of queries that are run is quite limited and can be initiated via a form. However, this will not likely be the case in the future for semistructured knowledge representations. Next I enumerate the situation in which the traditional form-based query specification is appropriate (i.e., rather than NL front-end):

1. A homogeneous domain includes the description of objects, identified by their names (e.g., cars, flowers, people).
2. A domain is completely unstructured; semantic links between its entities and objects are nonsystematic.
3. A domain is oriented to professional users and includes specific terminology.
4. Domain structure is very clear and a user is closely familiar with it.
5. The domain itself is almost unstructured, but the objects of search fall into clusters in accordance to their features. A Boolean combination of keywords is then well-suited for the search of objects in such a domain.

BACKGROUND

An NL front-end for a database implements query translation from NL to SQL. The difficulty in this task is that, first, NL is ambiguous and, second, its understanding requires domain knowledge that is not represented in a relational database (i.e., meanings of involved objects as words). The NL processing components that are required to achieve an accuracy desired under database querying are as follows (Galitsky, 2003; Gayatri & Raman, 2001; Wallace, 1984):

1. Morphological and syntactic analyses that produce links between words (i.e., a parser). This kind of analysis uses linguistic but not domain knowledge data.
2. Semantic analysis that establishes a mapping between some words or multiwords and table names, column names, and record values as well as SQL operators. Semantic analysis is bases on the formal treatment of meanings of involved entities (Allen, 1995).
3. Pragmatics analysis, which evaluates the consistency of the obtained query against the database and then filters the hypotheses of syntactic and pragmatic analyses in case an input query is ambiguous.
In a traditional architecture of database front-end, the components include an index, a lexicon, and a parser (Adam & Gangopadhyay, 1997). The index is used to uniquely identify each form in the system through a conceptual representation of its purpose. The form fields specify database or nondatabase fields whose values are either entered by the user (i.e., user defined) or are derived by the form (i.e., system defined) in response to user input. A set of grammar rules is associated with each form. The lexicon consists of all words recognized by the system, their grammatical categories, roots, their associations (if any) with database objects and forms. The parser scans a natural language query to identify a form in a bottom-up fashion. The information requested in the user query is determined in a top-down manner by parsing through the grammar rules associated with the identified form.

The enumeration of the features for a database NL front-end (About.com; Beck, Mobini, & Kadambari; Gayatri & Raman, 2001) follows:

- Independent domain creation: Given a database, a system may be capable of automatically adjusting its syntactic and semantic unit to it, processing the names of columns and tables. Indeed, automatic creation of entities which correspond to lexical units and relationships between them is quite unreliable for a natural language front-end to a database. This is in contrast to a natural language interface to a collection of documents (automatic annotation), which is quite reliable.

- Follow-up questions: For example, if one initially asks, Which store in California sold the most coffee in 1997? she can then ask a follow-up question, such as, Which one sold the most coffee? and the NL front-end system will understand this to mean Which store in California and In 1997 sold the most coffee.

- Specifying additional phrasings: This is necessary so that syntactically different but semantically similar questions are converted into the same SQL query.

- Maximum complexity of queries: These can be measured as a number of entities in a query, assuming that they are interconnected. If a question can be split into a conjunction or disjunction of simpler questions, a user is expected to do it. For a database front-end, a query complexity can be measured as a number of tables mentioned.

- Automatic superposition of syntactic and semantic templates: If a NL system knows two entities (or a respective pair of phrasings) it can combine it online to represent a query. This is a quite desirable feature to increase the complexity of queries and also to merge various databases each having its own front-end.

**TECHNIQUE OF SEMANTIC HEADERS**

The technique of semantic headers (SH; Galitsky, 2003) is intended to be the means of conversion of a relational database of abstract textual documents into a form, appropriate to be associated to a question and to generate an advice. There are two opposite common approaches to this problem. The first one assumes that complete formal representation of any textual document is possible, and the second one assumes that the textual information is too tightly linked to NL, and it cannot be satisfactorily represented without it. The former approach relies on the match of formalized query with the full-knowledge representation for answers, and the latter is based on the syntactic match between the question and sentences from answers. An important role of machine-learning-based technique for question answering is worth mentioning as well (Ng, Lai Pheng Kwan, & Xia, 2001).

The technique intermediate in respect to the degree of knowledge formalization. Only the data, which can be explicitly mentioned in a potential query, occur in semantic headers. The rest of the information, which would be unlikely to occur in a question but can potentially form the relevant answer, does not have to be formalized.

SH technique is based on logical programming, taking advantage of its convenient handling of semantic rules on one hand, and explicit implementation of the domain commonsense reasoning on the other hand. The declarative nature of coding semantic rules, domain knowledge, and generalized potential queries introduces logical programming as a reasonable tool. At the same time, the machinery of text annotation by the set of keywords has been proven to leverage the machine learning technique. Instead of using the keywords as semantic means to represent the meaning of a short textual document (answer), we use the logical formula where the keywords serve as atoms. Therefore, SH technique is a way of merging potential results of statistical approach to Q/A with the logical programming way of matching the formal representation of a query with the formal representation of an answer (semantic header of this answer). In legal domains, when the semantic of conversational language can only be ambiguously mapped into the semantic of the legal language, using just the statistical annotation by keyword does not lead to satisfactory results.

Consider the Internet auction domain, which includes the description of bidding rules and various types of auctions.

- **Restricted-Access Auctions:** This separate category makes it easy for you to find or avoid adult-only merchandise. To view and bid on adult-only items, buyers need to have a credit card on file.
Related Content

Application of Decision Tree as a Data Mining Tool in a Manufacturing System
www.igi-global.com/chapter/application-decision-tree-data-mining/7951?camid=4v1a

Knowledge and Object-Oriented Approach for Interoperability of Heterogeneous Information Management Systems
www.igi-global.com/article/knowledge-object-oriented-approach-interoperability/51219?camid=4v1a

IT Value of Software Development: A Multi-Theoretic Perspective
www.igi-global.com/chapter/value-software-development/4294?camid=4v1a

Mining for Mutually Exclusive Items in Transaction Databases
www.igi-global.com/chapter/mining-mutually-exclusive-items-transaction/8030?camid=4v1a