From Databases to Natural Language: The Unusual Direction

Yannis Ioannidis*

Dept. of Informatics & Telecommunications, MaD*g*IK Lab
University of Athens, Hellas (Greece)
yannis@di.uoa.gr
http://www.di.uoa.gr/∼yannis

Abstract. There has been much work in the past that combines the fields of *Databases* and *Natural Language Processing*. Almost all efforts, however, have gone in one direction: given unstructured, natural-language elements (requests, text excerpts, etc.), one creates structured, database elements (queries, records, etc.). The other direction has been mostly ignored, although it is very rich in application opportunities as well as in research problems. This short paper outlines several aspects of this other direction, identifying some relevant technical challenges as well as corresponding real applications.

Keywords: Databases, natural language processing, text synthesis.

1 Introduction

Any information systems environment, including one that involves database systems, may be abstracted as having three layers: its users, its functionality (realized through some form of a management system), and its content. Natural language may play some important role in the system functionality that is related to the two end layers: in the front-end, user interactions may be expressible in natural language; in the back-end, the original database content itself may actually be in natural language. Clearly, natural language is only one alternative available for each layer: in the front-end, user interactions may be (and usually are) performed using query languages, graphical or form-based user interfaces, and other dialog modes; likewise, in the back-end, content is usually structured or semi-structured, and is stored as relational or object-relational tables, XML or RDF documents, or some other form of objects.

In the past, there has been much research work on the interplay between natural languages and the other alternatives in each case above. Most of this work is on transforming natural language queries into some formal, structured query language, such as SQL. From very early on, researchers became excited about

* Partially supported by the European Commission under contract FP7-ICT-215874 "PAPYRUS: Cultural and Historical Digital Libraries Dynamically Mined from News Archives".

E. Kapetanios, V. Sugumaran, M. Spiliopoulou (Eds.): NLDB 2008, LNCS 5039, pp. 12–16, 2008.

this problem, and this has continued until today; there are numerous papers that address various aspects of the problem and several systems that have been developed to perform exactly such translations. Other user interactions, beyond queries (e.g., updates or constraints), have also been investigated in the past in the same way, although much more rarely.

More recently, there has been some significant activity on the content side as well. In particular, information extraction from natural-language text has been identified as an important area, and several efforts have been devoted into obtaining partial understanding of free text and generating database records, ontology relationships, or other (semi-)structured information that can be stored and manipulated by a database.

As one may easily observe from the above, most efforts of the past that combine the fields of *databases* and *natural language processing* have gone in one direction: given unstructured, natural-language items (requests, stored text documents, etc.), one creates structured, database items (SQL queries, relational records, etc.). The other direction has been mostly ignored, although it is very rich in application opportunities as well as in research problems. The two sections below motivate why the other direction is interesting as well. They describe some real applications, identify some relevant research problems, and outline some possible solutions. The first section deals with user interactions in the front-end, while the second section deals with the database contents, whether primary data inserted by users or metadata of various forms.

2 User Interaction Elements

Consider a schema with two tables:

$$\text{EMP}(\underline{\text{eid}},\text{sal},\text{age},\textit{did}) \text{ and } \text{DEPT}(\underline{\text{did}},\text{dname},\textit{mgr}),$$

where primary keys are underlined and foreign keys are in italics. Consider someone posing the following SQL query:

select e1.name **from** EMP e1, EMP e2, DPT d
where e1.did = d.did **and** d.mgr = e2.eid **and** e1.sal > e2.sal

There are several reasons why having the system provide a natural language interpretation of the query may be useful. Before the query is sent for execution, it may be nice for the user to see it expressed in a way that is most familiar, as verification that the query captures correctly the intended meaning. Seeing something like "Find the names of employees who make more than their managers" for the above query will be very helpful in making sure that this was indeed the user's original intention. The more complicated the query, the more important such feedback is.

In general, any situation where explanation of queries is warranted, such natural-language interpretation may be very useful and effective. For example, when a query returns an empty answer, it is nice to know the parts of the query that are responsible for the failure. Similarly, when a query is expected to return

a very large number of answers, it is useful to know the reasons, in case a rewrite would reduce the number significantly and would serve the user better.

Clearly, the same can be said about all other commands a user may give to a database system. Insertions, deletions, and updates, especially those with complicated qualifications or nested constructs, will benefit from a translation into natural language. Likewise for view definitions and integrity constraints, which borrow most of their syntax from queries. Also, although the example above was in SQL, similar arguments can be made about Relational Algebra queries, RDF queries in SPARQL or RQL, even Datalog programs, and others. One can claim that novice users may benefit by natural-language specification of even queries posed by filling out a form. Especially for large forms, where a user is likely to not know the underlying semantic connections among the fields presented in the form, a textual explanation may come in handy.

Needless to say, offering the functionality described above is not trivial for complicated queries and other commands. Part of the complexity lies with the fact that there are several alternative expressions of a query in a formal language that are equivalent, based on associativity, commutativity, and other algebraic properties of the query constructs. Capturing the query elements in the right order so that the corresponding textual expression is natural and meaningful independent of the way the user has expressed the query is not straightforward. Similarly, expressing queries with complex embeddings or aggregations is hard. For example, for the same schema above, consider the following two queries:

> **select** dname **from** EMP e, DEPT d **where** e.did=d.did **groupby** did
> **having count**(**distinct** sal)=1 **and count**(**distinct** age)=1
>
> **select** e.name **from** EMP e **where** e.sal $\leq$ **all**
> (**select** sal **from** EMP **where** did = e.did)

For a system to recognize that a good way to express the meaning of these, relatively simple queries is with phrases like "Find the names of departments whose employees all have the same salary and the same age" and "Find the names of employees with the lowest salary in their department" is nontrivial. Identifying the correct use of pronouns is one source of difficulties. Another one is related to whether or not the natural-language expression will be declarative (as in the above two examples) or procedural, i.e., whether it will just specify what the query answer should satisfy or also the actions that need to be performed for the answer to be generated. The former is always desirable, but for complicated queries, the latter may be the only reasonable approach. Identifying the complexity point where this becomes the case, however, is far from understood, and work must be done on this.

3 Database Contents

Similar issues arise in the back-end, when database contents are considered for translation into natural language. For example, consider a database that follows

the small schema presented earlier and suppose that one wants to have a textual description of its contents. If the database is large, it would make sense to create a textual summary of it, otherwise, a description of its full contents. There are several situations where such translation into natural language may be useful and desirable. Creating a short company description for a business plan or a bank-loan application or collateral material for marketing are some instances. Given other appropriate schemas, one can imagine textual descriptions in several other practical cases: a short description of a museum's exhibits, possibly customized to a visitor's particular interests; a brief history of a patient's medical conditions; the highlights of a collection in a digital library, with a few sentences on the main authors in the collection; a summary of a theater play in an information portal; and others.

Whatever holds for whole databases, of course, holds for query answers as well, especially those with some nontrivial structure, i.e., entire relational databases, complex objects, etc. Textual answers are often preferred by users, whether experienced or not, as they convey the essence of the entire query answer in an immediately understandable way. Moreover, they are critical for visually impaired or similarly disabled users, as they can be read to the user through a speech recognizer.

Clearly, the idea of translating data into natural language can be extended to all other forms of primary or derived data that a database may contain. Database samples, histograms, data distribution approximations are all, in some sense, small databases and can be summarized textually as above. Describing the schema itself, its basic entities, relationships, and other conceptual primitives offered by the model it is based on, is just a special case of a database description. User profiles maintained by the system for offering personalized answers, browsing indexes, and other forms of metadata are amenable to and may benefit from natural-language translation.

As with queries and other user-interaction elements, translating database contents to natural language is far from trivial. On one hand, it is simpler from translating queries, as the extent of alternative equivalent expressions of schemas and data is much narrower than that of queries. On the other hand, it is much more complicated than translating queries, as one has to choose the appropriate schema elements and data items that need to be captured in the textual summary. Furthermore, identifying the right linguistic constructs, introducing pronouns where appropriate, and synthesizing everything to produce a natural end result is equally complex. Although there has been some recent investigation of the topic, much more work is needed to devise an approach that is comprehensive, efficient, and effective at the same time.

4 Conclusions

In this short paper, we have looked into the intersection of the Database and Natural Language Processing areas and have outlined several interesting problems that arise when one attempts to translate database elements into natural

language elements, i.e., going in the opposite direction than usual. We have offered example applications that indicate the practical usefulness of the problem, have identified several categories of database elements whose translation into text would be useful, and have briefly described some of the technical challenges that need to be addressed in the future. We hope that researchers will take up this type of problems and help to push this interesting area forward.