Quelo: a NL-based intelligent query interface

Enrico Franconi, Paolo Guagliardo, and Marco Trevisan

franconi@inf.unibz.it, paolo.guagliardo@stud-inf.unibz.it, evenjn@gmail.com
KRDB Research Centre, Free University of Bozen-Bolzano, Italy

Introduction and Motivation

A controlled natural language (CNL) is a language engineered to look and feel like natural language and to be more suitable, for a specific purpose, than natural language itself. CNLs have been designed to improve communication between humans, like PoliceSpeak [8], to improve performance of machine translation, like ScaniaSwedish [8], and to edit and query knowledge bases, like Attempto [4] (e.g. as used in [2]). In this article we address CNLs of the latter kind.

According to a survey [1] on natural language interfaces to (query) databases (NLIDBs), one of the problems that specifically affect NLIDBs is the so-called "linguistic vs conceptual failure": when the system fails to answer a query, the user does not know whether it failed because the query could not be interpreted or because its interpretation was not supported by the schema of the database. A similar problem affects CNL-based interfaces as well: in this case, the system informs the user whether the query could be interpreted correctly, because the language is defined precisely, but—still—the user cannot tell whether the query retrieved no results because its interpretation was not consistent with the schema or simply because there was no data matching the requested features.

The described problem arises only when the user does not know the schema. In this situation, even if users are proficient with SQL or other formal query languages, they cannot query the database without knowledge of the schema. When the schema is largely unknown, users lack the vocabulary necessary to formulate queries in the first place. We must point out that CNL interfaces provide some help in such a situation, through so-called predictive editors. Thanks to a formal, unambiguous grammar and to a closed vocabulary, such an editor displays all syntactically legal completions of the user input at every single keystroke. This feature lets the user have a glance at the vocabulary of the language, which give a hint about the content of the database.

Nevertheless, even when the user knows the vocabulary supported, the user needs to know how the terms they can be combined meaningfully, and what constraints hold on the data, in order to formulate queries that the system can answer. For example, predictive editors that rely only on syntactical information do not stop users from composing a meaningless query like "*Which songs are performed by two songs?*", but also a query like "*Which albums have two artists?*", that is reasonable even if it fails when the schema associates artists with songs but not with albums. If that is the case, even if the latter query is syntactically correct, no data will ever match it. In this article we introduce the *Quelo* system, which addresses this very problem, by supporting the user in the task of formulating a precise query – which best captures their information needs – even when the user ignores completely the vocabulary and the constraints of the underlying information system holding the data.

Our idea develops a technique presented in earlier systems. "Conceptual authoring" [6] allows the user to compose a query by assembling snippets of text associated with semantic elements that are determined by the semantic context. NLMenu [9] uses semantic grammars to display the possible semantic extensions of a query in a menu. Both systems use a dedicated conceptual schema to guide the user during the composition of the query in the same way a predictive editor uses a syntactic grammar. These schemas are domain-dependent: when system engineers configure the system to query a new knowledge base, they must craft one such schema specifically for it. CNLs cannot take advantage of these techniques immediately, because these techniques were not designed as stand-alone reusable components.

We developed a novel system, named *Quelo* and formally described in [3], that exploits Description Logic (DL) *ontology schemas* to capitalise the service provided by semantic grammars, that is, filtering out semantically inappropriate query extensions and suggesting the appropriate ones given the context of the query and the background knowledge represented by the ontology schema. The purpose of Quelo is similar to purpose of the prediction subsystem of a syntax-based predictive editor, but Quelo works on the semantic level, and therefore it is not syntax-aware, and it is language independent. The improvement over the conceptual authoring systems developed so far consists in the use of a DL reasoner to enforce semantically consistency in the user's queries. Consider for example the following ontology schema, two queries and two query fragments. We present them in English using an informal paraphrase, but they would be best written using a first order logic syntax, or, even better, using DL syntax.

(A) "*there are only men and women*": for each X, if X is a person then (X is a man or X is a woman);
"*men don't love anyone*": there is no X such that a man loves X;
"*men don't hate each other*": for each X, if X is a man, no men hate X;
"*women hate only men*": for each X, if a woman hates X, X is a man.
(B) I am looking for a person X who hates a person Y who hates a man.
(C) X is a woman.
(D) X loves a person.
(E) I am looking for a married woman.

Given the schema A, our tool can avoid suggesting C and D as an extension of query B, because according to the constraints in A, X will always be a man in B. But Quelo goes even further: it drives the user in composing informative queries—that is to say—it tries to limit the insertion of redundant elements. A case in point: when Quelo retrieves the possible substitutions for "married" in query E, it will filter out "person", even if "person" is semantically compatible with "woman". The reason for this exclusion is that a "woman" is a "person" already, therefore "person woman" would not be more informative than "woman".

Query Representation and Reasoning

In a nutshell, Quelo works on a tree-shaped, conjunctive, description logics formula. It uses a DL reasoner to calculate which extensions would make the formula unsatisfiable with respect to a OWL DL ontology, in order to filter them out. In addition, it filters out some of the extensions that, if added, would produce an equivalent formula.

The system has a few, simple requirements. First, our "semantic grammar" is an OWL DL ontology schema. Such a schema is standard technology used to represent a formal conceptualisation of a knowledge domain, that is, a conceptual schema. If an OWL DL ontology Modelling the target knowledge base already exists, our tool needs no information besides the schema to provide its services. Second, the system requires a reasoner for OWL DL. Many such reasoners are developed commercially and available off-the-shelf under various licensing agreements.

The input query is a directed tree, where each edge is labelled with exactly one binary predicate and each node is labelled with one or more unary predicates. Such a tree encodes a first-order logic query in which the root node represents a free variable; each other node represents a distinct, existentially quantified variable; the label R of an edge directed from node x to node y represents a formula $R(x, y)$; each label C of a node x represents a formula $C(x)$. The overall formula encoded by the tree is the conjunction of all the formulae represented by edge and node labels and therefore it is a conjunctive query: no negations, no disjunctions, no universal quantifications may appear. Moreover, it is tree-shaped: no variables may appear twice as the second argument of a binary predicate. For instance, the query "I'm looking for a woman who loves a married man" corresponds to the following formula:

$$\exists y\, \mathrm{Woman}(x) \wedge \mathrm{loves}(x, y) \wedge \mathrm{Married}(y) \wedge \mathrm{Man}(y) \ .$$

The above restrictions on the shape of the query allow Quelo to deliver its services fast enough to provide them on-line, that is, at the same time the user types the query. Given such a query in input, the services offered by Quelo [5] consist in:

- retrieve additional labels that can be added to a given node: these are unary predicates that are not disjoint from the labels of the given node and are not equivalent to, more general than or more specific than any of them;
- retrieve the labels that can be associated with a newly created edge attached to a given node: these are binary predicates that, if added to the query, do not make it unsatisfiable;
- retrieve the labels that can be used to replace a given selected portion of the tree: these are unary predicates that are equivalent to, more general than or more specific than the formula represented by the selection and, if used in its stead, do not make the query unsatisfiable.

Each set of unary predicates included in the result is arranged in a hierarchy according to the constraints of the target ontology.

Natural Language Interface

In order to evaluate and showcase the system, we implemented a web-based natural language interface (NLI) [10], following the conceptual authoring paradigm [6] and using a custom-built natural language generation (NLG) module for English. Conceptual authoring is a technique to compose a textual document enriched with a precise, unambiguous specification of its meaning. In our NLI the document is a query: a textual representation (in English) of the user's informative need, enriched with a specification of its meaning — a labeled, directed tree query model.

To compose a query in the NLI, the user starts from an initial, system-defined sentence (”I am looking for something.”). At its margins (and, later on in the process, within it) there are gaps where the user can insert snippets of text chosen among those proposed by the system. Some parts of the sentence, determined by the system, can be deleted, or replaced with other similar snippets. The insertion, deletion and substitution of system-determined snippets are the only operations allowed. Through a sequence of such operations, the user edits the text. At the same time, hidden to the user, the system maintains a directed tree corresponding to the meaning of the query. Each operation on the text affects the tree by inserting, deleting or substituting a label, a node or a portion of the tree. Each part of the text is connected with a specific element of the tree, and each snippet that is inserted or removed carries in or out an element of the tree. In this way, the system builds the meaning of the text by observing the operations performed. In particular, the system does not take into consideration the text as a whole resulting from these operations in order to produce the tree. In this way, the system avoids the disadvantages associated with natural language understanding.

The snippets inserted and removed cannot be simple canned text, otherwise the result of their composition would hardly resemble English. For this reason, our interface uses NLG. Each of our snippets consists in a so-called NLG “template”, that is, a portion of a parse tree conforming to a simple English grammar with feature structures. Each such snippet has three layers: a surface textual representation, which is displayed to the user when it is suggested by the system, a portion of parse tree, which is used to produce the text as whole after the insertion of the snippet in the query, and a portion of a labeled directed tree, which represents the contribution of the snippet to the meaning of the query.

So far, the system we described follows closely the conceptual authoring archetype: the query is presented to the user in a generated natural language tightly connected with the underlying semantics. The ontology schema can thus be effectively exploited by the users in order to formulate a natural language query that best captures their information need, without any ambiguity. Our improvement on the basic conceptual authoring system is that the snippets available for addition and for replacement are filtered by Quelo. As explained in the second section Quelo operates by means of appropriate automated reasoning techniques over the ontology schema which describes the domain of the data in the information system. For each snippet, Quelo checks whether the portion of labeled directed tree of the snippet is a valid extension of the current query, effectively restricting the user’s choice to only those snippets which are relevant and meaningful in a given context.

The most powerful and innovative feature of the conceptual authoring interface of our system lies in the fact that users need not to be aware of the underlying organisation of the data, nor they need knowledge of the vocabulary, and still they tool will guide them into composing a query that is sound with respect to the underlying information system. Moreover, such knowledge can be gradually acquired by using the tool itself, gaining confidence with the vocabulary and with the constraints of the ontology. Users may also decide to just explore the knowledge without actually querying the information system, with the aim of discovering general information about the modelled domain.

To work properly, our NLI requires a map from elements of the ontology schema to such “templates”. To ease the adoption of the system, we devised a rule-based technique to extract linguistic information from the ontologies themselves in order to generate the NLG templates automatically. In order to achieve this result, we harvested and analysed more than 12000 unique schema relations, and we carefully selected the grammatical features supported by the generated natural language in order to keep the templates simple to build while being flexible enough to satisfy the most common expression needs of existing ontologies. While limited to English, the technique is effective to the point that the NLG interface can be used with most ontologies virtually without the need of human intervention. [10] [7]

Conclusions

We developed a natural language interface for Quelo, but Quelo was not specifically designed for it. In fact, we are currently developing other interfaces as well. We believe it could also be used within CNL-based predictive editors to limit the set of suggestions. Intuitively, when the query composed so far can be unambiguously transformed into a well-formed formula, it could be fed to Quelo in order to filter out suggestions associated with semantic extensions that are syntactically correct but not semantically appropriate. A complete description of this work together with an extensive set of references can be found in [3].

References

1. I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. Natural language interfaces to databases - an introduction. *Natural Language Engineering*, 1(01):29–81, 1995.
2. Abraham Bernstein, Esther Kaufmann, Anne Göhring, and Christoph Kiefer. Querying ontologies: A controlled english interface for end-users. In *Proceedings of ISWC 2005: The 4th International Semantic Web Conference*, pages 112–126, 2005.
3. Enrico Franconi, Paolo Guagliardo, and Marco Trevisan. An intelligent query interface based on ontology navigation. In *Proceedings of the Workshop on Visual Interfaces to the Social and Semantic Web (VISSW 2010)*, 2010.
4. Norbert E. Fuchs, Uta Schwertel, and Rolf Schwitter. Attempto controlled english - not just another logic specification language. In Pierre Flener, editor, *Logic-Based Program Synthesis and Transformation*, number 1559 in Lecture Notes in Computer Science, Manchester, UK, June 1999. Eighth International Workshop LOPSTR'98, Springer.
5. Paolo Guagliardo. Theoretical foundations of an ontology-based visual tool for query formulation support. Master's thesis, Free University of Bozen-Bolzano (Italy) and Vienna University of Technology (Austria), October 2009.
6. Catalina Hallett, Donia Scott, and Richard Power. Composing questions through conceptual authoring. *Computational Linguistics*, 33(1):105–133, 2007.
7. Laura H. Perez. Intelligent query interface: adding natural language support. Technical Report KRDB09-7, KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy, April 2009.
8. Jonathan Pool. Can controlled languages scale to the web? In *Proceedings of CLAW 2006: The 5th International Workshop on Controlled Language Applications*, 2005.
9. Harry R. Tennant, Kenneth M. Ross, Richard M. Saenz, Craig W. Thompson, and James R. Miller. Menu-based natural language understanding. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 151–158, Morristown, NJ, USA, 1983. Association for Computational Linguistics.
10. Marco Trevisan. A portable menu-guided natural language interface to knowledge bases for Querytool. Master's thesis, Free University of Bozen-Bolzano (Italy) and University of Groningen (Netherlands), January 2010.