Interactive Natural Language Query Construction for Report Generation*

Fred Popowich
School of Computing Science
Simon Fraser University
Burnaby, BC, CANADA
popowich@sfu.ca

Milan Mosny
Response42 Inc
North Vancouver, BC, Canada
Milan.Mosny
@response42.com

David Lindberg
School of Computing Science
Simon Fraser University
Burnaby, BC, CANADA
dll4@sfu.ca

Abstract

Question answering is an age old AI challenge. How we approach this challenge is determined by decisions regarding the linguistic and domain knowledge our system will need, the technical and business acumen of our users, the interface used to input questions, and the form in which we should present answers to a user's questions. Our approach to question answering involves the interactive construction of natural language queries. We describe and evaluate a question answering system that provides a point-and-click, web-based interface in conjunction with a semantic grammar to support user-controlled natural language question generation. A preliminary evaluation is performed using a selection of 12 questions based on the Adventure Works sample database.

1 Introduction

There is a long history of systems that allow users to pose questions in natural language to obtain appropriate responses from information systems (Katz, 1988; El-Mouadib et al., 2009). Information systems safeguard a wealth of information, but traditional interfaces to these systems require relatively sophisticated technical know-how and do not always present results in the most useful or intuitive way for non-technical users. Simply put, people and computers do not speak the same language. The question answering challenge is thus the matter of developing a method that allows users with varying levels of technical proficiency to ask questions using natural language and receive answers in an appropriate, intuitive format. Using natural language to ask these questions may be easy for users, but is challenging due to the ambiguity inherent in natural language anaylsis. Proposals involving controlled natural language, such as (Nelken and Francez, 2000), can deal with some of the challenges, but the task becomes more difficult when we seek to answer natural language questions in a way that is domain portable.

Before we can attempt to design and implement a question answering system, we need to address several key issues. First, we need to decide what knowledge our system needs. Specifically, we must decide what *linguistic knowledge* is needed to properly interpret users' questions. Then we need to consider what kind of *domain-specific knowledge* the system must have and how that knowledge will be stored and accessed. We must address the challenges posed by *users* with varying levels of technical sophistication and domain knowledge. The sophistication of the user and the environment in which the system is used will also affect how users will give *input* to the system. Will we need to process text, speech, or will a simpler point-and-click interface be sufficient? Finally, we must decide how to best *answer* the user's questions, whether it be by fetching pre-existing documents, dynamically generating structured database reports, or producing natural language sentences. These five issues do not present us with a series of independent choices that are merely stylistic or cosmetic. The stance we take regarding each of these issues strongly influences design decisions, ease of installation/configuration, and the end-user experience.

Here we solve this problem in the context of ac-

*This research was supported in part by a discovery grant from the Natural Sciences and Engineering Research Council of Canada. The authors would also like to thank the referees for their insights and suggestions.

INLG 2012 Proceedings of the 7th International Natural Language Generation Conference, pages 115–119,
Utica, May 2012.

cessing information from a structured database – a natural language interface to a database (NLIDB) (Kapetanios et al., 2010). However, instead of treating it as a natural language analysis problem, we will consider it as a task involving natural language generation (NLG) where users build natural language questions by making choices that add words and phrases. Using our method, users construct queries in a menu driven manner (Tennant et al., 1983; Evans and Power, 2003) to ask questions that are always unambiguous and easy for anyone to understand, getting answers in the form of interactive database reports (not textual reports) that are both immediate and consistent.

This approach retains the main advantage of traditional NLIDBs that allow input of a question in a free form text – the ability for the user to communicate with the information system in English. There is no need for the user to master a computer query langauge such as SQL or MDX. Many disadvantages of traditional free input NLIDBs are removed (Tennant et al., 1983). Traditional NLIDBs fail to analyze some questions and indicate so to the user, greatly decreasing the user's confidence in the system. The problem is even worse when the NLIDB analyzes the question incorrectly and produces a wrong or unpexpected result. In contrast, our system is able to answer every question correctly. In traditional free input NLIDBs, the user can make grammatical or spelling mistakes that may lead to other errors. Using a menu-based technique, the user is forced to input only valid and wellformed queries. The complexity of the system is greatly reduced as the language that the system has to process is simple and unambiguous. Portability to other domains is improved because there is no need for vocabulary that fully covers the domain.

2 Our approach

We begin with an overview of our approach to this question answering problem involving NLG. We describe how we address each of the afore-mentioned issues and give our rationale for each of those choices. Following a brief discussion of our use of online analytical processing (OLAP) (Janus and Fouche, 2009) in section 2.2, we then decribe how we use the OLAP model as the basis for interactive natural query generation, and describe the database used in our evaluation, along with the grammar used for NLG.

2.1 Overview

Our approach to the question answering problem is based on the following decisions and assumptions:

Linguistic knowledge We use a semantic grammar to support user-controlled NLG rather than language analysis. By guiding the construction process, we avoid difficult analysis tasks, such as resolving ambiguities and clarifying vague language. We also eliminate the possibility of out-of-domain queries.

Domain-specific knowledge We model domain knowledge using an OLAP cube, a widely-used approach to model domain-specific data. OLAP cubes provide a standard semantic representation that is well-suited to historical business data and allows us to automatically generate both the lexicon and the semantic grammar for our system.

Users The prototypical user of our system is familiar with business issues but does not have a high-degree of technical expertise. We provide a simple and intuitive interface suitable for such users but still powerful enough for users of any level of technical proficiency.

Input A web-based, point-and-click interface will guide users in the creation of a natural language query string. Users click on words and phrases to construct a question in plain English.

Answers We will answer questions with an interactive database report. Users can click on parts of the report to get detailed information, making it more of an interactive dashboard rather than a report.

An approach governed by these principles offers many benefits. It simplifies database report creation and lowers the associated costs, allows businesses to leverage existing investments in data warehouse and reporting technology, offers a familiar and comfortable interface, does not require installation on client machines, and is simple to install and configure.

2.2 Role of OLAP

An OLAP cube is produced as a result of processing a datawarehouse into datastructures optimized

for query processing. The OLAP query language makes reference to measure groups (that roughly correspond to fact tables), measures (that come from the numerical values in the fact tables) and dimensions (that come from dimension tables). For example, the *order* fact table might include total order price, order quantity, freight cost, and discount amount. These are the essential figures that describe orders, but to know more we need to examine these facts along one or more dimensions. Accordingly, the dimension tables associated with this fact table include time (order date, year, quarter, and month), customer (name, address, city, and zip code), and product (name, category, and price).

2.3 Interactive Natural Language Generation

At the heart of the system is a semantic grammar. Our goal was to create a grammar that is suitable to database querying application, but is simple enough so that it can be automatically adapted to different domains. The semantic model makes use of both entities (unary predicates) and relationships (binary predicates) that are automatically derived from the OLAP model. These entities and relationships can be directly and automatically mapped to the lexical items and phrases that the user sees on the screen during query construction. Once a user has completed the construction of a natural language query, a corresponding first order logic formula is created which can then be translated into a database query in SQL or MDX.

Our assumption was that many database queries can be expressed within the following template

```
Show <Show> and ... and <Show> for
each <GroupBy> and ...  and for
each <GroupBy> limit to <LimitTo>
and ... and  to <LimitTo>
```

where <Show>, <GroupBy> and <LimitTo> are different classes of nominals. <Show> may refer to a measure or to a level in a dimension which may take an additional constraint in a form of a prepositional clause. <GroupBy> may refer to a level in a dimension which may take a constraint in a form of a prepositional phrase or to a set of members of a dimension. <LimitTo> may refer to a set of members of a dimension. A prepositional phrase expressing a constraint has a form

```
with <NounPhrase>
```

```
QuestionElement
   Terminal
      EntityTerminal
         GroupByEntityTerminal
   Nonterminal
      TopLevel
      GroupBy
      LimitTo
      Show
      PrepositionalClause
      Determiner
      NounPhrase
      List
```

Figure 1: Semantic Grammar Element Classes

where the noun phrase consists of a determiner such as “some”, “no”, “at least N”, “exactly N” and a noun referring to a measure.

The semantic grammar makes use of classes in an inheritance hierarchy as shown in Figure 1. Each question element corresponds to a parametrized terminal or nonterminal. That is, it can play a role of one of multiple terminals or nonterminals depending on its initialization parameters. There are altogether 13 classes that comprise the elements of the grammar. The implementations of the different class elements make use of semantic constraints as appropriate. Only minimal human intervention is required when adapting the system to a new OLAP cube. The intervention consists of “cleaning up” the automatically generated terminal symbols of the semantic grammar so that the plural and singular forms that were present in the cube metadata are used consistently and so that the mass vs. countable attribute of each measure is set appropriately.

3 Evaluation

An evaluation of this kind of system requires an examination of three performance metrics: domain coverage, ease of use, and query efficiency. How well the system covers the target domain is crucially important. In order to measure domain coverage, we need to determine how many answerable questions can actually be answered using the system. We can answer this question in part by examining the user interface. Does the interface restrict users’ access to domain elements and relationships? A more thorough assessment of domain coverage requires exten-

sive user studies.

Ease of use is often thought of as a qualitative measure of performance, but a systematic, objective evaluation requires us to define a quantitative measure. The primary action used to generate queries in our system is the "click." Users click on items to refine their queries, so the number of clicks required to generate queries seems like a reasonable starting point for evaluating ease of use. The time it takes users to make those clicks is important. A four-click query sounds efficient, but if it takes the user two minutes to figure out which four clicks need to be made, not much is gained. It would be ideal if the number of clicks and the time needed to make those clicks grow proportionally. That is, we do not want to penalize users who need to build longer queries.

Query efficiency is measured by the time between the user submitting a query and the system presenting the answer. How long must a user wait while data is being fetched and the report generated? Unlike ease of use, this is objectively measurable and easy to benchmark.

In our initial evaluation, we applied these metrics to a selection of 12 natural language questions about the data in the Adventure Works (Codeplex Open Source Community, 2008) database that could be answered by our natural language query construction system. These questions were generated by a user with prior exposure to the Adventure Works database but no prior exposure to the query construction software system or its design or algorithms, so the questions are not purposely fine-tuned to yield artificially optimal results. Eight of these questions were directly answerable, while four were indirectly answerable. For each of these questions, we measured the number of clicks required to generate the query string, the time it took to make the required clicks, and the time required to retrieve the needed records and generate a report. The distinction between directly answerable and indirectly answerable questions deserves a short explanation. A question is deemed directly answerable if the answer is the sole result returned in the report or if the answer is included in a group of results returned. A question is deemed indirectly answerable if the report generated based on a related query can be used to calculate the answer or if the information relevant to the answer is a subset of the information returned. So, the question *What are the top 20 products based on internet sales* was directly answerable through the constructed query *Show products with one of 20 highest internet sales amount*, while the question *What is the averagefreight cost for internet orders over $1000* could only be answered *Show internet freight cost for customers with more than 1000 dollars of internet sales amount and for each date*.

We found that a user was able to construct natural language queries using between 2 and 6 clicks which required 10 and 57 seconds of elaspsed time for the construction process. On average 3.3 clicks were required to create a query with an average time of 33 seconds, where the time grew in a linear manner based on the number of clicks. Once a query was constructed, the average time to generate a report was 6.7 seconds with the vast majority of queries producing a report from the database system in 4 seconds or less. The median values for query construction was 2.5 clicks, query construction was 31.5 seconds, and report generation was 4 seconds..

4 Analysis and Conclusions

Our evaluation suggests that the menu driven NLG approach results in the rapid creation of unambiguous queries that can retrieve the relevant database information corresponding to the query. It has been embedded in a system that uses OLAP cubes to produce database reports (and dashboards) that allow user interaction with the retrieved information. The system was automatically adapded to a given OLAP cube (only minimal human intervention was required) and can be equally easily adapted to other OLAP cubes serving other domains.

Our results build on semantic web related work (Paiva et al., 2010) that shows that use of NLG for guided queries construction can be an effective alternative to a natural language interface to an information retrieval system. We deal with a highly constrained natural language (cf. the analysis grammars used by (Nelken and Francez, 2000; Thorne and Calvanese, 2012)) that is effective in generation of database queries and the generation (not analysis) of natural language. Like (Paiva et al., 2010), we rely on a semantic grammar, but instead build on the information that can be automatically extracted from the database model, rather than leveraging knowl-

edge from semantic web resources. Furthermore, we provide a more detailed evaluation as to the effectiveness of the guided query construction technique.

Use of OLAP in NLG has also been explored in the context of content planning (Favero and Robin, 2000), and can play an important role in dealing with domain portability issues not only in the context of NLG but also in other natural language database applications. Our technique for leveraging the data model and OLAP cube avoids human customization techniques like those reported by (Minock, 2010) where an explicit mapping between phrases and database relations and entities needs to be provided, and (Evans and Power, 2003) where explicit domain information needs to be entered.

The NLG query construction approach does have limitations, since users will likely have questions that either cannot be constructed by the semantic grammar, or that cannot be answered from the underlying database. However, issues related to choice or ambiguity that are frequently encountered by NLG systems in particular, and natural language processing systems in general, can be avoided by having a human "in the loop."

Efficiency and effectiveness is derived from how we leverage human knowledge, both in query composition and result interpretation. In traditional, non-intelligent query scenarios, users know what they want to ask but not necessarily how to ask it. By guiding the user through the NLG process, the user can focus on the *what* not the *how*. Database reports are generated quickly, providing unambiguous answers in a clear, flexible format. and in a familiar, comfortable, un-intimidating web-based environment. Aside from usability benefits, this web-based approach has the added benefit of minimizing configuration and maintenance.

Our results are only suggestive, since they involve only 12 questions. They suggest it would be worthwhile to expend the resources for a full study that includes multiple users with different levels of experience, multiple domains and larger sets of questions. A more fine-grained analysis of the difference between the results sets of constructed English queries and the expected answers to original questions should also be performed along with an evaluation of how easy it is for the user to find the answer to the question within the database report.

References

Codeplex Open Source Community. 2008. *Adventureworks SQL Database Product Samples*. CODEPLEX. http://msftdbprodsamples.codeplex.com.

Faraj A. El-Mouadib, Zakaria S. Zubi, Ahmed A. Almagrous, and Irdess S. El-Feghi. 2009. Generic interactive natural language interface to databases (GINLIDB). *Int Journal of Computers*, 3:301–310.

Roger Evans and Richard Power. 2003. WYSIWYM - building user interfaces with natural language feedback. In *Proc. of EACL 2003, 10th Conf. of the European Chapter of the ACL*, pages 203–206, Budapest, Hungary.

Eloi Favero and Jacques Robin. 2000. Using OLAP and data mining for content planning in natural language generation. In *NLDB '00 Proc. 5th International Conference on Applications of Natural Language to Information Systems-Revised Papers*, pages 164–175. Springer-Verlag, London.

Phil Janus and Guy Fouche. 2009. Introduction to olap. In *Pro SQL Server 2008 Analysis Services*, pages 1–14. Springer-Verlag.

Epaminondas Kapetanios, Vijayan Sugumaran, and Myra Spiliopoulou. 2010. Special issue: 13th international conference on natural language and information systems (NLDB 2008) five selected and extended papers. *Data and Knowledge Engineering*, 69.

Boris Katz. 1988. Using english for indexing and retrieving. In *Proceedings of the First RIAO Conference on User-Oriented Content-Based Text and Image Handling (RIAO '88)*. CID.

Michael Minock. 2010. C-PHRASE: a system for building robust natural language interfaces to databases. *Data and Knowledge Engineering*, 69:290–302.

Rani Nelken and Nissim Francez. 2000. Querying temporal databases using controlled natural language. In *Proc 18th International Conference on Computational Linguistics (COLING 2000)*, pages 1076–1080, Saarbrücken, Germany, August.

Sara Paiva, Manuel Ramos-Cabrer, and Alberto Gil-Solla. 2010. Automatic query generation in guided systems: natural language generation from graphically built query. In *Proc 11th ACIS Intl Conf on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2010)*, pages 165–170. IEEE Conf Publishing Services.

Harry Tennant, Kenneth Ross, Richard Saenz, Craig Thompson, and James Miller. 1983. Menu-based natural language understanding. In *Proc 21st annual meeting of the Association of Computational Linguistics*, pages 151–158. ACL.

Camilo Thorne and Diego Calvanese. 2012. Tractability and intractability of controlled languages for data access. *Studia Logica, to appear*.