Information Extraction for SQL Query Generation in the Conversation-Based Interfaces to Relational Databases (C-BIRD)

Majdi Owda, Zuhair Bandar, and Keeley Crockett

The Intelligent Systems Group, School of Computing, Mathematics and Digital Technology, The Manchester Metropolitan University, Chester Street, Manchester, M1 5GD, UK
{m.owda,z.bandar,k.crockett}@mmu.ac.uk

Abstract. This paper presents a novel methodology of incorporating Information Extraction (IE) techniques into an Enhanced Conversation-Based Interface to Relational Databases (C-BIRD) in order to generate dynamic SQL queries. Conversational Agents can converse with the user in natural language about a specific problem domain. In C-BIRD, such agents allow a user to converse with a relational database in order to retrieve answers to queries without knowledge of SQL. A Knowledge Tree is used to direct the Conversational Agent towards the goal i.e. creating an SQL query to fit the user’s natural language enquiry. The use of IE techniques such as template filling helps in answering the user’s queries by processing the user’s dialogue and extracts understandable patterns that fills the SQL templates. The developed prototype system increases the number of answered natural language queries in comparison to hardcoded decision paths in the knowledge trees.

Keywords: Information Extraction, Knowledge Trees, Conversational Agents, and Conversation-Based Interfaces to Relational Databases.

1 Introduction

The development of reliable Natural Language Interfaces to Databases (NLIDBs) can accelerate the progress of interactive applications. A considerable amount of literature has been published on this area. Androutsopoulos [21] defined four main approaches to NLIDBs, each approach is supported by a specific architecture.

The first approach is based on Pattern-Matching which is one of the first architectural techniques that have been used for the design and development of NLIDBs. An example of such a system is SAVVY [22]. In order to make NLIDBs more reliable this approach utilizes a highly structured semantic grammar, rapid authoring tools, and a query paraphraser. However, in large databases a great number of patterns have to be created. Furthermore, the process of identifying relevant patterns is critical and time consuming. The second approach is based on an Intermediate Language, which is used to represent an intermediate logical query generated from a natural language question. The intermediate logical query is then transformed into SQL. Two such
systems of this category are MASQUE/SQL [5] and EDETE [3]. The main disadvantage of this approach is that failure to generate an SQL query will result in a response which is not informative to the user.

The third approach is based on a Syntax-Based Family of Architectures, where the natural language question is syntactically analyzed to create parse tree(s). The resulting parse tree(s) will then be used directly to create database queries. An example of such a system is LUNAR [8]. However, a major difficulty of this approach is in generating the mapping rules from the parse tree to SQL. The fourth approach is based on the Semantic-Grammar Family of Architectures. This approach also involves the construction of a parse tree and mapping of the parse tree to SQL. The difference between this approach and syntax based approach is the use of semantic categories instead of syntactic concepts where non-leaf nodes in the parse tree are semantic categories. The semantic information about the knowledge domain is hard-wired into the semantic grammar. This family includes systems such as: PLANES [7, 21] and LADDER [21]. This approach appears to work well in restricted domains (e.g. small databases). However, creating new system will involve creating a new semantic grammar which is both time consuming and complex.

Currently there is some ongoing work on NLIDBs which does not fall into the categories defined by Androutsopoulos such as Precise [1], Step [6] and C-Phrase [19]. A reliable NLIDB system should overcome the problems associated with the above approaches. Firstly any such system should provide adequate coverage of patterns to represent the domain. Secondly, the system should allow for interactivity with the user. The user must be able to understand the system responses in the case of failure of generating SQL query. Thirdly, overcoming the problem of generating mapping rules from syntax based parse tree to SQL.

The Conversation-Based Interfaces to Relational Databases [1] is based on the pattern-matching architecture but has additional components: knowledge trees, a goal oriented conversational agent and information extraction module. Knowledge trees are used to structure the domain knowledge to cover common queries in a particular domain. Whereas a goal oriented conversational agent provides the natural language interface and can help to disambiguate the user’s queries, by allowing for dialogue interaction. The use of the knowledge trees helps direct the conversational agent towards the goal (i.e. query generation from natural language) through interaction with the user. The information extraction module helps extract understandable patterns from the relational database and project the created templates into SQL queries. The proposed approach reduces the complexity of building NLIDBs.

This paper is organized as follows: Sections 2, 3 and 4 will introduce the main concepts behind Conversational Agents, Knowledge Trees, and Information Extraction. Section 5 will introduce the Enhanced Conversation-Based Natural Language Interfaces to Relational Databases Framework. Section 6 presents the results. Section 7 includes the summary.

2 Conversational Agents

A conversational agent is human-computer dialogue system that interacts with the user turn by turn using natural language. Conversational agents have been used in