Natural Language Interface to Relational Database (NLI-RDB) through Object Relational Mapping (ORM)

Mr Abdullah Alghamdi, Dr. Majdi Owda and Dr. Keeley Crockett

School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Chester Street, Manchester, M1 5GD, UK
Email: abdullah.a.alghamdi@stu.mmu.ac.uk, {m.owda, k.crockett}@mmu.ac.uk

Abstract This paper proposes a novel approach for building a Natural Language Interface to a Relational Database (NLI-RDB) using Conversational Agent (CA), Information Extraction (IE) and Object Relational Mapping (ORM) framework. The CA will help in disambiguating the user's queries and guiding the user interaction. IE will play an important role in named entities extraction in order to map Natural Language queries into database queries. The ORM framework i.e. the Hibernate framework resolves the impedance mismatch between the Object Oriented Paradigms (OOP) and Relational Databases (RDBs) i.e. OOP concepts differ from RDB concepts, thus it reduces the complexity in generating SQL statements. Also, by utilizing ORM framework, the RDBs entities are mapped into real world objects, which bring the RDBs a step closer to the user. In addition, the ORM framework simplify the interaction between OOP and RDBs. The developed NLI-RDB system allows the user to interact with objects directly in natural language and through navigation, rather than by using SQL statements. This direct interaction tends to be easier and more acceptable for humans whom are nor technically orientated and have no SQL knowledge. The NLI-RDB system also offers friendly and interactive user interface in order to refine the query generated automatically. The NLI-RDB system has been evaluated by a group of participants through a combination of qualitative and quantitative measures. The experimental results show good performance of the prototype and excellent user's satisfaction.

Keywords: Natural Language Interfaces to Relational Databases (NLI-RDB); Object Relational Mapping (ORM); Natural Language Interfaces; Database Interfaces and Hibernate Framework.

1 Introduction

In the era of information technology, smart phones and big data, the demand for retrieving information from databases in a short time is an urgent need, in particular for decision-makers who are normally inexperienced in structured query language (SQL). In addition, there are many aspects of our lives that can benefit a technology

that provides interfaces to relational databases such as applications on smartphones and mini hand-held devices (e.g., tablets). Moreover, such technology allows people to access databases using natural language (e.g., English) [1, 2, 3, 4, 5, 6]. Therefore, there is an increasing demand on querying the RDBs through the use of natural language instead of having experts in SQL. This requires developing Natural Language Interfaces to Relational Databases (NLI-RDBs). The main challenge that developers of NLI-RDBs continue to face, is how to map or convert the natural language questions into SQL queries automatically [1, 2, 3,4].

NLI-RDBs research is an important area since 1960s and are still an interesting research field. In the literature, several NLI-RDB architectures have been proposed for developing prototype systems [1, 2, 6, 7, 8, 9, 10], which can be grouped in five main architectures: Pattern-Matching Architecture, Intermediate Representation Language Architecture, Syntax-Based Architecture, Semantic Grammar-Based Architecture and Intelligent Agents-Based Architecture [1, 2, 4]. The first approach is based on Pattern-Matching, which is one of the earliest architectures that used for developing NLI-RDBs. This approach uses checking mechanisms to check whether a set of tokens, as a user input, matches a pre-defined pattern. Based on matched tokens, a specific SQL statement is executed [4]. The Pattern-Matching approach is simple but does not require complex natural language processing for implementation [1, 2]. SAVVY [11] is an example of the Pattern-Matching Architecture. The second approach is based on an Intermediate Language in which the user's question is translated from natural language into a high-level representation language independent of database (intermediate representation language). Then, the intermediate representation language is converted into SQL queries using database query generator [1, 2, 4]. MASQUE/SQL [10] and EDETE [12] are examples of using the Intermediate Representation Language Architecture. Independence from the database is the main advantage of this system where it can be applied in different domains and databases [4]. The third approach is a Syntax-Based Architecture in which the natural language query is syntactically analyzed to create parse trees that are used directly to create database queries [1, 2]. LUNAR [13] is an example system of the Syntax-Based Architecture. The fourth approach is a Semantic-Grammar Architecture, which is largely similar to the Syntax-Based Architecture, but it used semantic categories instead of syntactic concepts. It depended on semantic grammar rules to parse the user inputs into semantic parse tree, which is then mapped into SQL query [1, 2, 4]. LADDER [2, 4] and PLANES [1, 2] are two examples of semantic-based NLI-RDB systems. The fifth approach is the Intelligent Agents-Based Architecture. This approach is containing knowledge and information about the user and the surrounding world. This knowledge made such systems able to handle questions, which could not be directly responded to using database management systems. In addition, it provided better understanding of the user's needs [1, 2]. LOQUI [2] is an example of Intelligent Agents-Based systems. Whilst, each approach has advantages and disadvantages, the common major drawbacks are either with the lin-

guistic and conceptual challenges i.e. natural language understanding and processing pitfalls or the database interaction component i.e. presenting an understandable responses to user in the case of failure to generate SQL query and in addition to offer the user to interact with the results in case of successful mapping from natural language to SQL query [3].

The NLI-RDB novel architecture proposed in this paper built upon three main pillars; firstly a Pattern Matching Conversational Agent (PM-CA) component; in which it is used to for offering friendly interaction with the user and partly resolving the interaction drawback highlighted in previous work; secondly, an Information Extraction (IE) component which offers the capability of extracting named entities in real time in which will help in answer the user query. Thirdly, the key novelty of the proposed NLI-RDBs, which is the usage of the ORM framework to offer better interaction with RDBs through visualizing RDBs entities into real world objects. Therefore offering better interaction with user in the case of linguistic, conceptual or conversation failure with the conversational agent; in which the interaction with RDBs entities is a step closer to the user i.e. more understandable for the user to have navigable objects created in real time in response to their queries in which this will provide an easier way for interaction compared to SQL statements or un-understandable error messages in case of failure to generate SQL query automatically.

This paper is organized as follows: sections 2, 3 and 4 will introduce the main concepts in Information Extraction (IE), Conversational Agent (CA) and Object-Relational Mapping (ORM). Section 5 will introduce the challenges in developing NLI-RDBs. Section 6 will introduce the proposed NLI-RDB Framework. Section 7 will present the evaluation results. Finally, section 8 provides the conclusion of this study.

2 Information Extraction (IE)

IE task is primarily for extracting structured information from unstructured or semi-structured sources. The pioneers in IE are Andersen et al. in late 1970 [14] used IE technique for extraction of facts from press releases to generate news stories. An IE methodology consist of several stages, and at each stage the method will add a structure and often lose information, hopefully irrelevant, by applying rules that are acquired manually and/or automatically. In the literature, there are two main approaches to design an IE methodology. The first is the knowledge engineering approach, which is based on having a knowledge engineer to develop rules for the IE methodology. According to Appelt et al. [15] the knowledge engineering based approach is most effective when resources such as lexicons and rule writers are available. The second approach is the automatic training approach, which does not require a knowledge engineer, instead, it only requires someone who knows well the domain, and then the task is to annotate a corpus of texts for the information being extracted. An IE method can be a part of various systems such as Knott et al. [9], which uses IE to analyses financial discussion boards for automated crime detection. Owda et al. [16] incorporate IE techniques into an Enhanced Conversation-

Based Interface to Relational Databases (C-BIRD) in order to generate dynamic SQL queries. In Owda et al work IE played an important role in the named entities extraction, which added unique successful features for the conversational agent in order generate SQL queries in real time. In addition, IE has been successfully used in many other fields such as Web Knowledge Bases [17], Text Mining [18] and bioinformatics [19].

3 Conversational Agent (CA)

Conversational Agent (CA) is a computer system which can employ text to allow people to communicate with computer systems using natural language. The pioneer in this field is Alan Turing who discussed the question: "Can machines logically process information?" [20]. Since then CA interfaces have been used effectively in many applications such as customer service, help desk, website navigation, technical support [21], web-based guidance [22], tutoring [23], assessment and training [24] and database interfaces [1, 2, 25]. There have been a number of CAs used as an interface to relational database such as C-BIRD [1, 9], which allows a user to converse with a relational database in order to retrieve answers to queries without knowledge of SQL. The C-BIRD methodology combines pattern-matching conversational agent with knowledge trees and information extraction modules. Similarly, Choudhary et al. [26] propose another pattern matching approach in which patterns have been created for simple query, aggregate function, relational operator, short-circuit logical operator and joins. All in all; CAs proved to be useful component in building NLI-RDBs.

4 Object-Relational Mapping (ORM)

Relational Databases Management Systems (RDBMSs) have been used on a large-scale as an ideal solution for storing and retrieving data [5]. Object Oriented Programming (OOP) has been used as a major vehicle for developing interfaces to the RDBMSs for human friendly information retrieval and manipulation purposes. The OOP has been used to develop many NLI-RDBs [1, 9, 25]. However; a number of challenges have been highlighted in the literature such as impedance mismatch i.e. OOP concepts does not directly match with RDBMS concepts and OOP developers are not usually RDBMSs developers; therefore it is difficult to handle and manipulate data stored in RDBMSs using OOP i.e. through SQL interaction. ORM has arisen trying to bridge the gap of mismatch between RDBs and OOP [27]. ORM is a programming technique that provides a solution for OO systems to exchange data with RDBs safely and smoothly by passing entities from and to the relational database through objects as needed. The ORM framework was developed in an open source project for Java programmers in 2002, which is called the Hibernate framework, established by Gavin King [28, 29]. Consequently, systems that use this technology started to appear (e.g., ADO.NET Entity Framework [28] from Microsoft).

In the programing field, developers prefer to deal with persistent data existing in programing objects instead of dealing with SQL statements for accessing data in

databases, even though this could lead to “impedance mismatch” between tabular data and object state. However, ORM frameworks, such as Hibernate, reduce the gap of mismatch between OOP and RDBs by converting data from RDBs into appropriate programing objects and vice versa. The ORM framework sits between applications’ i.e. object world and RDBs. It plays the role of mediator for mapping object schemas to database schemas [28].

5 Challenges in Developing NLI-RDBs

The challenges in developing NLI-RDBs can be summarized as follow:

- **Linguistic versus Conceptual Failures**
  Linguistic and conceptual failures refer to the inability of the NLI-RDB system to process natural language query. When the system fails to answer users’ questions, the users try to re-write their questions thinking that the problem refers to the linguistic coverage, whereas it is caused by conceptual failure or vice-versa [2].
- **SQL Query Generation Failure**
  SQL is a cumbersome language for normal users [30]. Generating SQL statement automatically through mapping from natural language questions into SQL queries; is a complex process. In addition to the challenge in carrying out further interaction after the failure to generate SQL query and presenting an understandable response during this failure.
- **Ambiguity**
  Ambiguity is one of the main challenges for developing NLI-RDBs [2, 4, 31], which can be present at lexical level, syntax level and referential level [32]. Lexical ambiguity refers to the ambiguity in words. (e.g., a word can be a noun or verb depending on the context). Syntax ambiguity refers to the interpretation of one sentence in various ways [32]. Referential ambiguity is present when entities mentioned previously implicitly or explicitly; then can be referred to these entities at later stage as pronouns, possessive determiners or noun phrases [2].
- **Users Assume Intelligence**
  NLI-RDBs must use techniques for identification and disambiguation of data and meta-data within the natural language context because users usually assume the system has intelligence (ability to understand all types of questions [2, 31]). For example, "Give me the location of ABC." where it is not specified whether ABC is a person, place or city name [31].

6 Methodology and Implementation

This section describes the creation of a NLI-RDB using CA, IE and ORM framework. The key features of the approach are shown in figure 1. The proposed architecture consists of five major components (User Interface, Text preparation, Engine Algorithm, ORM Framework and Relational Database). Each component contains

a set of modules. The following are explanations of the five components that will be introduced from top to the bottom of the figure for simplicity.

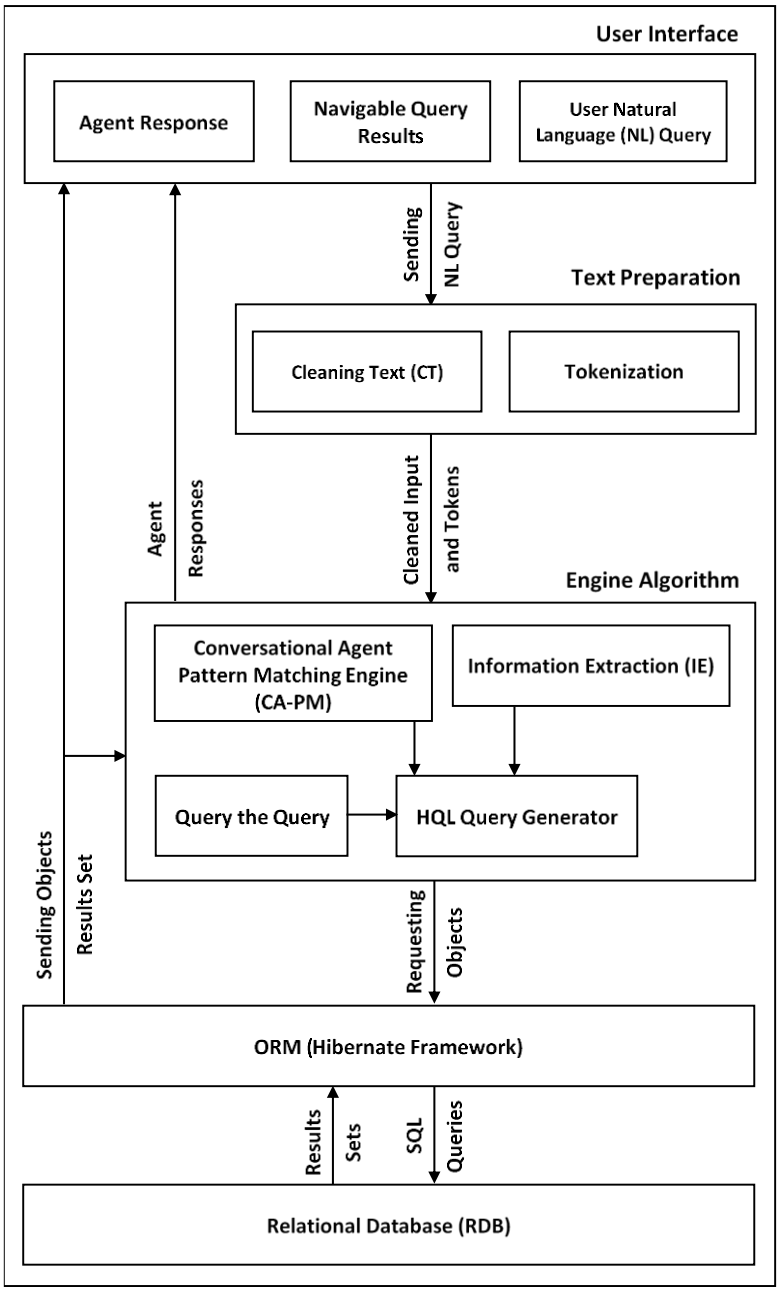

Fig. 1. The NLI-RDB system architecture

6.1 User Interface

The user interface provides interaction between the user and the application. Figure 2 shows the NLI-RDB prototype system user interface implementation which consists of three main parts.

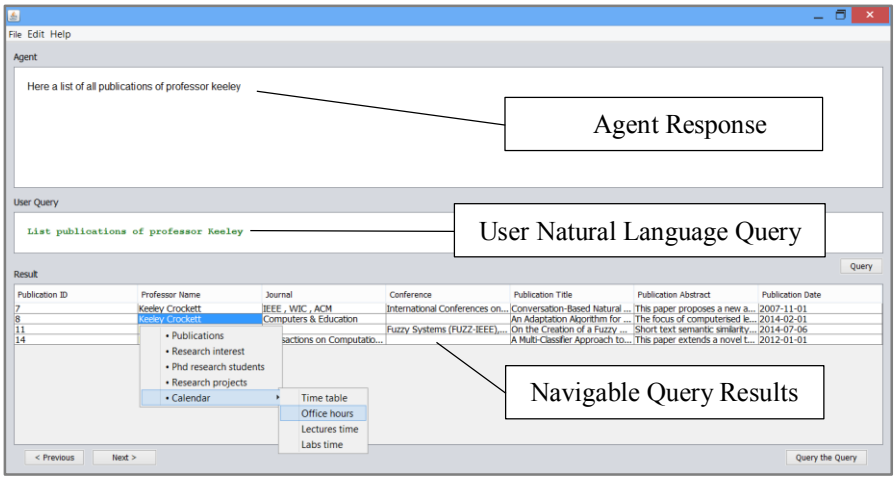

Fig. 2. The NLI-RDB system's user interface

- **User Natural Language Query**
  The user is offered to write his question in natural language such as English. The natural language query is sent to the next process to prepare it for analyzing.
- **Agent Response**
  The conversational agent displays conversational responses to the user and some useful comments regarding the user's query. The responses are based on pattern matching rules performed on algorithm engine unit.
- **Navigable Query Results**
  The results of queries are displayed as a navigable table, which includes simple and navigable menus. This offers various options for performing a query within a query by just a few clicks, without the need of writing again.

6.2 Text Preparation

This component prepares the text for analyzing by passing it through two main steps:

- **Cleaning Text (CT)**
  Before doing a textual analysis, often a text cleaning stage is required. This step cleans the user's query from unnecessary characters and words (e.g., white spaces, newlines, dashes, slashes, "could", "please", "a", "the", etc.).

- **Tokenization**
  Tokenization is the process of splitting a string of text into a set of separated words called tokens. The user's natural language query will be split into a sequence of tokens for further processing.

6.3 Engine Algorithm

The engine algorithm is the main component responsible for analyzing the user's natural language query in order to generate agent responses and Hibernate Query Language (HQL) statements. It comprises of four modules.

- **Information Extraction (IE)**
  IE component is responsible for filtering the user's query in order to extract the objects' names and their attributes, which represent the relational database tables and columns. The extracted information is then sent to HQL generator to be used for filling the HQL query template.
- **Conversational Agent Pattern Matching Engine (CA-PM)**
  CA-PM component is working to match the cleaned text and tokens with predefined patterns either in order to guide the user through conversational responses or to recognize objects attributes and possible conditions, which also will be used with HQL generator.
- **HQL Query Generator**
  HQL generator is responsible for building a suitable HQL query based on the extracted information and matched patterns. The generated HQL query is processed by the Hibernate ORM framework in order to generate the requested information from relational database as objects.
- **Query the Query**
  This is a set of predefined HQL templates to perform a query within a query using a Graphical User Interface (GUI) context menu. The users can perform other queries based on the result of their previous query using mouse clicks, without the need for writing natural language query again. The menu contents vary with respect to the query result. This provides an easy and fast interactive method to facilitate querying within a query result i.e. interacting with objects contents directly. This is shown in figure 2.

6.4 ORM Framework

The ORM framework (Hibernate) maps an object-oriented model to a relational database. The Hibernate establishes and manages communication with the database. The generated HQL query is translated by Hibernate into SQL query, which is then executed among the relational database. The obtained data is saved as persisted objects, which can be easily manipulated within the application. The objects results set is sent to the user interface to present the result for the user's natural language query.

6.5 Relational Database (RDB)

The Relational Database (RDB) is holding the domain information. The domain is designed for university professors and their details such as names, gender, job title and departments they are working in, as well as contact details. In addition, it holds information related to their research interests, projects, publications and PhD students. Moreover, it stores details about the academic timetable and office hours for each individual professor. The domain database has been created and populated for the purpose of this project and future research to be conducted.

6.5.1 The Projection of Relational Database Entities into Objects

The proposed NLI-RDB architecture mainly emphasizes the concept of projection of objects from the RDB tables. As a result, RDBs are brought a step closer to the user by mapping RDBs entities into real world objects. Figure 3 shows a demonstration sample of objects projected from RDB tables, which is easier for the user to understand and interact with. The database layer shows the relational database and its tables. It could be any RDBMS. The ORM layer represents the Hibernate ORM framework, which manages and persists data between Java objects and relational databases. The Hibernate ORM framework resolves the problem of object-relational impedance mismatch between OOP and RDBs, therefore it reduces the complexity in generating SQL statements during developing applications. The Objects layer shows objects have been generated from the relational database tables. The generated objects can be easily manipulated within the application and easier interact with by the user.

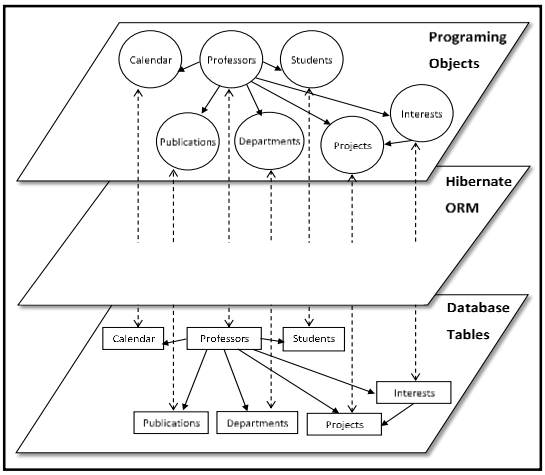

Fig. 3. The projection of domain objects generated from the relational database tables

7 Evaluation Results and Discussion

This section presents the evaluation methods that have been used in order to evaluate the NLI-RDB system prototype produced. The evaluation included both quantitative and qualitative measures. A questionnaire of two parts firstly, task success

[33] using six scenarios to measure system's ability of detecting tables and columns that answer the user's query; and secondly the system's usability [34] using seven questions to measure the system ease of use and interface appearance. The evaluation sample is based on using 15 participants in which seven were computer science students and eight from different fields. The participants were given an instruction sheet including a description for prototype system in addition to the evaluation sheet. They were given a typical time of 20 minutes to use the prototype system and complete the evaluation. In the first part of evaluation i.e. task success, the participants were asked to do six scenarios. Each scenario has a set of tasks that have a specific goal to be accomplished. They were asked to respond to each task whether the system gave the required information correctly or not, (answer 'Yes' or 'No'). In the second part, they were asked to give their overall opinion on the system by answering a questionnaire about the prototype usability. The measure's structures, details and results are described in the following subsections.

7.1 Quantitative Measures

In the Quantitative Evaluation, a task success [33] has been used to measure user's ability to complete specific tasks. The tasks have been divided into six scenarios based on the goals listed in table 1. Each scenario has a set of tasks that have a specific goal to be accomplished. It has been requested to answer 'Yes' or 'No' (whether the system gave the required information correctly or not).

Table 1. Task Success Scenarios

No.	Goal of the Scenario with Examples of Tasks
1	To measure retrieving information based on one table without a condition. 1. List all professors in the database. 2. List all professors' publications.
2	To measure retrieving information based on one table with a condition of one or more columns. 1. Tell me about professor Majdi. 2. What is the timetable of "enterprise programming" unit.
3	To measure retrieving information based on two tables with a condition of one column. 1. Who is supervisor of student Pei Lee? 2. List all publications of professor Majdi.
4	To measure retrieving information based on two tables with a condition of two or more columns. 1. List males professors in Mathematics department. 2. When is the annual leave of Keeley.
5	To measure retrieving information based on some numerical functions such as less than, greater than and between. 1. List papers published since 2012. 2. List all papers published between 2010 and 2013.
6	To measure retrieving information based on the functionalities of bar menus, context menus and "Query the Query". 1. Select professor "Keeley Crockett" and use right click to get her publication. 2. Use File menu to save the result as PDF file.

7.2 Qualitative Measures

In the Qualitative Evaluation, usability testing [34, 35, 36, 37, 38] has been done in order to measure users' usability satisfaction. Table 2 shows qualitative measures besides their goals. In addition, table 3 shows the qualitative measures scales.

Table 2. Qualitative measures

Measure	Goal
System is easy to use	To measure user satisfaction toward the prototype in terms of simplicity, flexibility and friendliness.
Understanding the system responses	To measure user satisfaction toward the prototype agent response in terms of clarity and content.
Using of context menu	To measure user satisfaction toward the prototype in terms of simplicity and flexibility of using context menus.
Using of top menu bar	To measure user satisfaction toward the prototype in terms of clarity and simplicity of using menu bar.
Text clearness	To measures user satisfaction toward the prototype in terms of type and size of used text.
Organization of information	To measures user satisfaction toward the prototype in terms of the information distribution in the interface.
General appearance	To measures user satisfaction toward the prototype in terms of general appearance of the interface.

Table 3. The qualitative measures scales

Measure	Scales		
System is easy to use	☐ easy	☐ moderate	☐ difficult
Understanding the system responses	☐ easy	☐ moderate	☐ difficult
Using of context menu	☐ easy	☐ moderate	☐ difficult
Using of top menu bar	☐ easy	☐ moderate	☐ difficult
Text clearness	☐ very good	☐ good	☐ poor
Organization of information	☐ very good	☐ good	☐ poor
General appearance	☐ very good	☐ good	☐ poor

7.3 Quantitative Results

The quantitative evaluation was carried out, in Scenarios 1 to 6. In each scenarios, the participants were asked to respond whether or not the prototype delivered the required information for each task, (answer ‘Yes’ or ‘No’). Table 4 shows the overall success of each scenario.

Table 4. The overall success of each scenario

Sce.	Goal of the scenario	Result
1	To measure the success of retrieving information based on one table without any condition.	100%
2	To measure the success of retrieving information based on one table with a condition of one or more columns.	98.89%
3	To measure the success of retrieving information based on two tables with a condition of one column.	93.33%
4	To measure the success of retrieving information based on two tables with a condition of two or more columns.	92%
5	To measure the success of retrieving information based on some numerical functions such as less than, greater than and between.	100%
6	To measure the success of retrieving information based on the functionalities of bar menus and context menus.	100%
The overall average		97.37%

7.4 Qualitative Results

In the qualitative evaluation, the participants were asked to provide their opinions about the ease of use of and the general appearance of the prototype. Based on the evaluation scales, results have been presented into two groups as shown in table 5 and table 6.

Table 5. The evaluation results of usability (group 1)

The qualitative measures (group 1)	easy	moderate	difficult
System is easy to use	93.33%	6.67%	0%
Understanding the system responses	80 %	20 %	0%
Using of context menu	86.67%	13.33%	0%
Using of top menu bar	100%	0.00%	0%

Table 6. The evaluation results of usability (group 2)

The qualitative measures (group 2)	very good	good	poor
General appearance	85.71%	21.43%	0%
Text clearness	92.86%	14.29%	0%
Organization of information	92.86%	7.14%	7.14%

The overall evaluation results showed an excellent user satisfaction in both the quantitative and qualitative evaluations used.

8 Conclusion

This paper proposed building a Natural Language Interface to a Relational Database (NLI-RDB) through using Conversational Agent (CA), Information Extraction (IE) and Object Relational Mapping (ORM) frameworks. The novelty of work is through introducing ORM framework in order to resolve the impedance mismatch between the Object Oriented Paradigms (OOP) and Relational Databases (RDBs) which reduces the complexity in generating SQL statements. In addition, by mapping the RDBs entities into real world objects, the RDBs entities are brought closer to the user. Additionally, the ORM frameworks simplify the interaction between OOP and RDBs. The developed NLI-RDB prototype system allows for the user to interact with objects directly, rather than by using SQL statements. This direct interaction tends to be easier and more acceptable for humans. The prototype also offers friendly and interactive user interface in order to refine the query generated automatically. The experimental results showed good performance of the prototype and excellent user's satisfaction.

References

[1] M. Owda, Conversation-based interfaces to relational databases (C-BIRDs), Manchester: Manchester Metropolitan University, 2012.

[2] I. Androutsopoulos, G. D. Ritchie and P. Thanisch, "Natural Language Interfaces to Databases – An Introduction," Natural Language Engineering, vol. 1, no. 01, pp. 29-81, 1995.

[3] N. Nihalani, S. Silakari and M. Motwani, "Natural language Interface for Database: A Brief review," International Journal of Computer Science Issues, vol. 8, no. 2, 2011.

[4] N. Sangeeth and R. Rejimoan, "An Exhaustive Study on NLIDB Systems and their Challenges," International Journal of Computer and Advanced Engineering Research (IJCAER), vol. 02, no. 02, 2015.

[5] K. Shabaz, J. O'Shea, K. Crockett and A. Latham, "Aneesah: A Conversational Natural Language Interface to Databases," in Proceedings of the World Congress on Engineering, London, 2015.

[6] M. Llopis and A. Ferrández, "How to make a natural language interface to query databases accessible to everyone: An example," Computer Standards & Interfaces, vol. 35, no. 5, pp. 470-481, 2013.

[7] X. Yiqiu, W. Liwei and Y. Shi, "The Study on Natural Language Interface of Relational Databases," in Environmental Science and Information Application Technology (ESIAT), 2010 International Conference, Wuhan, 2010.

[8] A. Shah, J. Pareek, H. Jyoti and N. Panchal, "NLKBIDB - Natural Language and Keyword Based Interface to Database," International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1569-1576, 2013.

[9] M. Owda, Z. Bandar and K. Crockett, "Information Extraction for SQL Query Generation in the Conversation-Based Interfaces to Relational Databases (C-BIRD)," in Agent and Multi-Agent Systems: Technologies and Applications, Springer Berlin Heidelberg, 2011, pp. pp 44-53.

[10] I. Androutsopoulos, G. R. P and Thanisch, MASQUE/SQL: An Efficient and Portable Natural Language Query Interface for Relational Databases, University of Edinburgh, Department of Artificial Intelligence, 1993.
[11] T. Johnson, "Natural language computing: the commercial applications," The Knowledge Engineering Review, vol. 1, no. 3, pp. 11-23, 1986.
[12] P. ReisAffiliated, J. Matias and N. Mamede, "Edite – A Natural Language Interface to Databases: a New Dimension for an Old Approach," in Information and Communication Technologies in Tourism, Springer Vienna, 1997, pp. pp 317-326.
[13] W. A. Woods, R. Kaplan and a. N.-W. B. , "The Lunar Sciences Natural Language Information System: Final report," Bolt, Beranek and Newman, 1972.
[14] P. M. Andersen, P. J. Hayes, A. K. Huettner, L. M. Schmandt, I. B. Nirenburg and S. P. Weinstein, "Automatic Extraction of Facts from Press Releases to Generate News Stories," in Proceedings of the third conference on Applied natural language processing, 1992.
[15] E. Applet and D. Israel, "Introduction to Information Extraction," in Tuotorial at IJCAI conference, 1999.
[16] M. Owda and E. Knott, "extraction, The detection of potentially illegal activity on financial discussion boards using information," in Conference: 2nd International Conference on Cybercrime, Security and Digital Forensics, 2012.
[17] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam and S. Slattery, "Learning to extract symbolic knowledge from the World Wide Web," in Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence, 1998.
[18] R. J. Mooney and R. Bunescu, "Mining knowledge from text using information extraction," in ACM SIGKDD Explorations Newsletter - Natural language , 2005.
[19] R. Bunescu, R. Ge, R. J. Kate, E. M. Marcotte, R. J. Mooney, A. K. Ramani and Y. W. Wong, "Comparative experiments on learning information extractors for proteins and their interactions," Artificial intelligence in medicine, vol. 33, no. 2, pp. 139-155, 2005.
[20] A. M. Turing, "Computing machinery and intelligence," Mind, vol. 59, no. 236, pp. 433-460, 1950.
[21] J. Lester, K. Branting and B. Mott, "Conversational Agents," in Practical Handbook of Internet Computing, New York: Chapman & Hall, 2004, pp. 220-240.
[22] A. Latham, K. Crockett and Z. B, "A conversational expert system supporting bullying and harassment policies," in In the 2nd ICAART, 2010.
[23] S. D'Mello, B. Lehman, J. Sullins, R. Daigle, R. Combs, K. Vogt, L. Perkins and A. Graesser, "A Time for Emoting: When Affect-Sensitivity Is and Isn't Effective at Promoting Deep Learning," in Intelligent Tutoring Systems, Springer Berlin Heidelberg, 2010.
[24] P. G. Kenny and T. D. Parsons, "Embodied Conversational Virtual Patients," in Conversational Agents and Natural Language Interaction: Techniques and Effective Practices, Information Science Reference, 2011, pp. 254-281.
[25] K. Pudner, K. Crockett and Z. Bandar, "An Intelligent Conversational Agent Approach to Extracting Queries from Natural Language," in World Congress on Engineering 2007 (Volume 1);2007, p305, 2007.
[26] N. Choudhary and S. Gore, "Pattern based approach for Natural Language Interface to Database," International Journal of Engineering Research and Applications, Vols. Vol. 5,, no. Issue 1(Part 2), pp. pp.105-110, 2015.
[27] G. King and C. Bauer, Java Persistence with Hibernate, Revised ed., Greenwich, CT: Manning Publications, 2006.
[28] E. O'Neil, "Object/Relational Mapping 2008: Hibernate and the Entity Data Model (EDM)," Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp. 1351-1356, 2008.
[29] C. Bauer and G. King, Hibernate in Action, Greenwich: Manning Publications Co., 2005.
[30] P. P. Filipe and N. J. Mamede, "Databases and Natural Language Interfaces," JISBD, pp. 321-332, 2000.

[31] A. Mohite and V. Bhojane, "Challenges and Implementation Steps of Natural Language Interface for Information Extraction from Database," International Journal of Recent Technology and Engineering (IJRTE), vol. 3, no. 1, pp. 108-111, March 2014.
[32] "AI - Natural Language Processing," Tutorials Point, 14 April 2015. [Online]. Available: www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_natural_language_processing.htm. [Accessed 5 August 2015].
[33] J. Nielsen, "Success Rate: The Simplest Usability Metric," 2001. [Online]. Available: https://www.nngroup.com/articles/success-rate-the-simplest-usability-metric/. [Accessed 10 December 2015].
[34] "Usability Evaluation Basics," usability.gov, 8 July 2013. [Online]. Available: www.usability.gov/what-and-why/usability-evaluation.html. [Accessed 5 October 2015].
[35] "Usability in Software Design," 2000. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms997577.aspx. [Accessed 2015 October 7].
[36] J. Nielsen, "10 Usability Heuristics for User Interface Design," 1 January 1995. [Online]. Available: www.nngroup.com/articles/ten-usability-heuristics. [Accessed 8 October 2015].
[37] D. J. Mayhew, "The Usability Engineering Lifecycle," CHI'99 Extended Abstracts on Human Factors in Computing Systems. ACM, pp. 147-148, 1999.
[38] J. Nielsen, "How Many Test Users in a Usability Study?," 4 June 2012. [Online]. Available: www.nngroup.com/articles/how-many-test-users/. [Accessed 3 October 2015].