Application of a Configurable Keywords-Based Query Language to the Healthcare Domain

Edgars Rencis
Institute of Mathematics and Computer Science, University of Latvia, Riga, Latvia
Email: Edgars.Rencis@lumii.lv

***Abstract*—Organizations tend to gather information more and more rapidly nowadays. The increase in the volume of this information is exponential. For it to be used efficiently, it is imperative to have a means for extracting valuable knowledge from the vast amount of provided information. The extraction of knowledge is considered the central part of a successful organization. Such an extraction must be done quickly and efficiently in order for this knowledge to be put in use at once. This paper addresses the problem of how this extraction of knowledge can be done conveniently without obtaining specific and in-depth skills in information technology so that it could be done by domain experts directly. The paper inspects the domain of hospital management with the domain experts being hospital managers and physicians who need to make decisions that would be based on facts gained from the information located in the database of the hospital. The system described in this paper lets users formulating their queries in the natural language, and it translated those queries in the language understood by the underlying database. Our case studies suggest that such type of querying might provide a noticeable improvement in the decision-making process of healthcare professionals.**

***Index Terms*—query language, hospital management, knowledge extraction, natural language processing, healthcare informatics, keywords-based**

I. Introduction

The information gathered by any organization increases more and more rapidly nowadays. All organizations accumulate various kind of information in their databases, but the most of this information may never be used to increase the quality of the operation of the organization or to alleviate the decision-making process by the domain experts. In order for the information to generate an actual benefit for the organization, the domain experts need to be able to exploit this information in their everyday work. To do this, one must be able to turn the information into knowledge, that is, one must be able to extract the knowledge from the vast volume of the available information. The knowledge can then be used to alleviate the decision-making process by the domain experts.

What are the main obstacles that prevent domain experts from extracting knowledge from the available information? To be able to perform the extraction process one must first be able to understand what kind of information is available in the databases of the organization, i.e. what is the structure of the database, what kind of data is stored in the database, how these data are connected etc. In a typical case the data is stored in a relational database, so to be able to answer the abovementioned questions one must delve deep into the ER model of the database and into all its technical details. The structure of the database if often very complicated for the domain experts that are not IT specialists, and it is, therefore, challenging for them to understand even the very first part of the knowledge extraction process, that is – what kind of data are located in the database of the organization.

Secondly, if the domain expert has understood the type of knowledge that could be extracted from the database, he/she must be able to formulate the queries in the language that is understandable by the database. If it is the relational database, the queries must be made using the SQL. In most cases, this language is far too complicated for the domain experts, so they have to turn to an IT specialist who would then help them write their queries in SQL. Also, the answers obtained from the system would be revised together with the IT specialist in order for them to be understandable by the domain experts.

It can be foreseen that the abovementioned process is not only very time-consuming but also very error-prone because between the two main parties – the system and the end-user – another party in the form of an IT specialist appears giving rise to additional errors. As a result, the decision-making process delays and becomes cumbersome, and the advancement of the organization is not optimal.

In this paper we propose to get rid of the involvement of the third party between the system and the end-user in the processes of understanding the structure of the database, formulating the queries and understanding their answers. To fulfil this objection, we developed a new kind of data ontologies – the Semistar Data Ontology [1] – that has proven to be very easily perceptible data format from the end-user point of view [2]. Based on this new data structure, we then developed the base query language with a very efficient implementation. Its initial goal was to become the query language for the end-users. However, our experiments with the healthcare practitioners showed that we had achieved this goal only partially – the language is indeed very well readable, but

Manuscript received August 8, 2020; revised February 3, 2021.

doi: 10.12720/jait.12.2.142-147

the domain experts are still not able to write queries in this language very well [1]. The Base Language is briefly described in Section III, and the performed case study is outlined in Section IV.

To solve this problem with the formulation of queries, we developed a system that allows user to write queries in keywords-contained natural language. These queries are then translated automatically into valid queries in our existing base query language. It can happen though that there are more than one possible translation results of the given query. Therefore we introduced the concept of the entropy to be able to range the translation results based on their probability of being the right ones. The keywords-based query language is inspected in Section V.

Next, we introduced the possibility for the user to choose one of the provided translation results that correctly describes the question that was in his/her mind. We strongly believe the domain expert can do that because the base query language has proven to be very well readable. Section VI provides insight into the concepts of entropy and penalties, which are the base concepts needed to evaluate the query translation results. The system then learns from the choices the user makes during the process and adapts to the words and phrases the user typically makes. As a result, the probability of correctly translating the query increases in future.

Finally, the default behavior of the system can be altered by configuring the penalties of various primitive operations performed on the original query during its translation process. This process is outlined in Section VII.

II. Related Work

The de facto standard of querying the data is nowadays the SQL language [3]. However, the way data and queries are described in the SQL is too complicated for end-users. There are similar languages for data querying in other types of data storage, e.g. SPARQL for RDF ontologies [4]. The SPARQL, as well as SQL, requires an accurate formulation of the query (both semantics and syntax) and solid skills in the underlying technology. Thus it makes a definition of queries too complex to learn and use. In order to reduce the complexity wrappers have been made for SQL and SPARQL, e.g. a graphical query builder "Graphical Query Designer" for SQL Server, a graphical query builder "ViziQuer" [5] for SPARQL and RDF databases, a form-based tool which uses standard GUI elements (e.g. tables and lists) and wizards "SAP Quick Viewer SQVI".

Another way of making querying easier is to base the querying onto the natural language. Many solutions exist there that are based on the assumption that we can use natural language for formulating queries. The natural language can be more- or less-strongly controlled. A Natural Language Interface to Databases (NLIDB) is developed in various sources. This interface can then be used to query relational databases [6]-[10].

We can also combine the natural language-based query language with a graphical user interface so that one can build a part of the query graphically while parts of it may be written textually. The biggest flaw of purely graphical languages is their lack of expressiveness. It can be mitigated by introducing the abovementioned parts in a textual language in which one could write the most complicated chunks of the query. Some examples of this approach are the NaLIR [11] and the DataTone [12] tools.

It must be noticed that usually most of the abovementioned related work is based exclusively on the English language or in some cases – on another prominent language. In this paper, we put the focus on developing the tool for the necessity of the hospitals of Latvia. Thus the system would be based on the Latvian language, which is used by the local domain experts in Latvia. The Latvian language is very different from English. From the morphological typology point of view, the English language is analytic, while the Latvian language belongs to the synthetic language family. It means that in English, the relationships between words in sentences are conveyed by way of so-called helper words (e.g. particles or prepositions) and also by word order. Meanwhile, in the Latvian language, the relationships are provided by adjusting the form of a word to communicate its role in the sentence. It is not therefore always possible to interchange the principles that are valid for synthetic and analytic languages.

III. The Base Query Language

The base query language (further in the text – The Base Language) is based on the natural language, and it was developed taking into account the same basic principles that serve as the basis of the SQL, i.e. there are several types of sentences (called the sentence templates) providing the structure of the queries. The sentence templates define the level of control of the underlying natural language, and they contain places where the users can enter their own words, expressions or other allowed linguistic structures.

The sentence templates are inspected thoroughly in the literature [1], [2], [13]-[17]. In order for this paper to be understood without delving deep into the technical nuances of the implementation of the Base Language, we will only provide some examples of valid queries in the Base Language. The structure of the sentences can be guessed from these examples.

COUNT Patients WHERE EXISTS HospitalEpisode WHERE referringPhysician = familyDoctor.

SUM totalCost FROM HospitalEpisodes WHERE dischargeReason=healthy AND birthDate.year() = 2012.

SELECT FROM HospitalEpisodes WHERE dischargeReason=deceased ATTRIBUTE responsiblePhysician.surname ALL DISTINCT VALUES.

The experiments with the Base Language showed that the language is very well readable by domain experts who understands the concepts of the underlying data ontology [1]. The problem, however, is that the language is still too strict for it to be easily used for writing queries. Although

it is based on the natural language, the level of control is still too high. That is the main reason why there originated the need for a less-strongly controlled natural language for building queries.

IV. Usability of the Base Language: A Case Study

To verify the usability of the Base Language, we performed a case study involving the end-users from the target group of the language – students of the Faculty of Medicine, University of Latvia. The language came together with a tool that allowed to write queries in the Base Language and to execute them. The execution of the query means that the tool inspects the underlying database (that is prepared beforehand to be in the form of the Semistar data ontology) and extracts the necessary information from the database. The tool then shows the extracted results of the provided query back to the user. The user can then refine the query if necessary and try to obtain new results. This kind of interactivity between the user and the tool allows for an easier and more convenient way of extracting valuable knowledge from the underlying database.

The whole process with the experiment was divided into two parts – the learning part and the doing part. The learning part consisted again of two parts. Firstly, the students were taught the basics of the tool. To be able to work with the tool, they also needed to have the necessary knowledge about the underlying data ontology – both from the structural and content point of view. They were briefly introduced to the concepts of the ontology, the class, the attribute, the association. The concepts contained by the particular ontology were already familiar to the students because the database was devoted to the domain of hospital management (see Fig. 1).

After having understood the underlying data ontology, the learning of the tool was straightforward because the tool is very simple and provides only the basic functionality needed for entering queries and obtaining their results.

The second part of the learning involved learning the Base Language. Students were taught the basic idea behind the language, i.e. that it is based on the natural language and that there are several types of sentences that can be made. Each type of sentences contains the fixed part in the form of the keywords and the structure of the sentence and the variable part where the users can insert their own specifics – the concepts from the underlying ontology and other types of items. This part of the teaching was based on examples so that the students would be able to understand the language more easily.

The whole learning process took time of about two hours. Afterwards, the second part of the experiment began where students had to perform tasks that involved exploiting the Base Language and the tool. There were two types of tasks – the reading task and the writing task.

The reading task was devoted to researching how readable is the Base Language. Students were given valid queries written in the Base Language, and they had to reformulate those queries into sentences in the natural language and to explain in their own words what kind of information would be extracted from the database by this particular query. The provided queries covered all types of the sentence templates of the Base Language, and they varied in their complexity.

The writing task was devoted to researching the aspect of how well one can formulate valid queries in the Base Language. Students were given sentences in the loose natural language that were not valid queries in the Base Language, and they had to write valid queries in the Base Language that would extract the asked knowledge from the database. In most cases, the given sentences were formulated in the manner that was quite far from the structure of valid queries. Also in this part of the experiment, the provided sentences covered all types of the sentence templates of the Base Language, and they varied in their complexity.

The results of the experiment were quite different for both types of tasks. The results of the reading task were very satisfactory. The vast majority of the students were able to answer most of the questions correctly, i.e. they were able to understand the queries written in the Base Language. However, the results of the writing task were not so good. Only three students out of 15 were able to write the requested queries correctly, while the majority of the students were able to write correctly only 25-90% of the queries (see Table I).

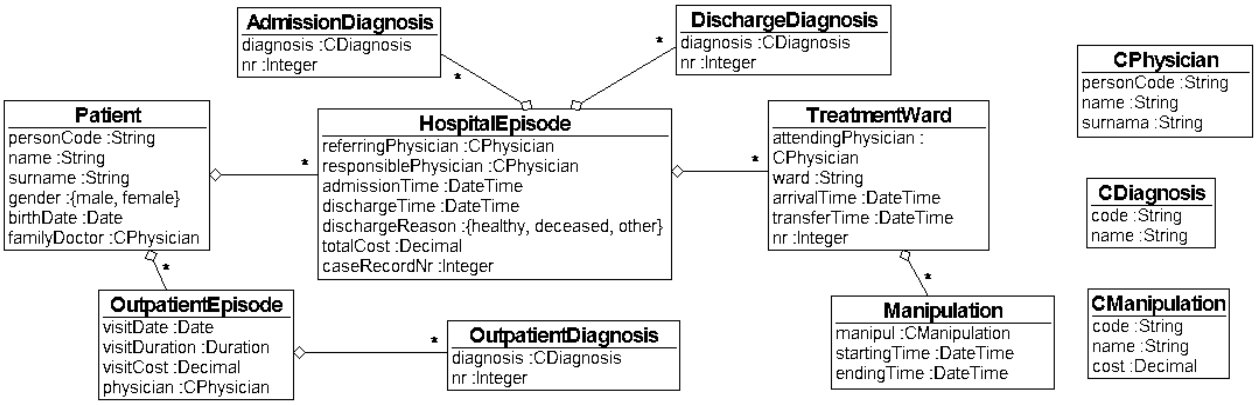

Figure 1. Semistar data ontology of the hospital management domain exploited in the case study.

TABLE I. RESULTS OF THE EXPERIMENT

	Number of students succeeded (n=15)				
Task execution level (%)	≥90	≥75<90	≥50<75	≥25<50	<25
The reading task	9	2	2	1	1
The writing task	3	5	2	3	2

The results show that the Base Language is very well readable by the domain experts. It is most likely thanks to two reasons – firstly, the Base language is based on the natural language and it, therefore, contains sentences that are natural to the reader, and, secondly, the insertions into the existing sentence templates are mostly concepts from the ontology that are familiar to the domain experts.

However, the results also suggest that the Base Language is not yet quite suitable for writing queries. It is due to the fact that despite the Base Language being based on the natural language, the level of control of the Base Language is quite high, and it demands a particular way of thinking for regular users to be able to formulate their thoughts accurately.

The main takeaway from the experiment is the conclusion that the level of control of the language must be lowered in order for it to be used as the end language in which the queries can be written, but at the same time, the Base Language can very well serve as an intermediate language for reading purposes.

V. KEYWORDS-BASED QUERY LANGUAGE

The main idea behind the less-strongly controlled natural language is to allow the user writing queries in the natural language that only contains particular keywords, but not forcing the user to exploit a particular structure of the sentences. The system could then extract the keywords from the provided query and to build a valid query in the Base Language from those keywords.

For example, the user could write such a sentence in the natural language:

Can you, please, find for me those patients who have been hospitalised at least 2 times?

This sentence is not, of course, a valid sentence in the Base Language because it does not conform to any of the sentence templates of the Base Language. However, it is still possible to deduce that the user has actually meant to write such a sentence in the Base Language:

SHOW Patient WHERE (COUNT HospitalEpisode) >= 2.

This deduction can be made by finding the keywords in the original query and by trying to combine them in ways that form a correct sentence in the Base Language. Such keywords are in this case the word "find" (a synonym of the keyword "show" of the Base Language), the word "patients" (the class name "Patient" from the underlying data ontology), the word "who" (a synonym or the keyword "where" of the Base Language), the word "hospitalised" (which could mean either the phrase "exist Hospital" or the phrase of a type "count (HospitalEpisode) > x") and the phrase "at least" (which denotes the comparison operator ">=").

It must be noticed that there can be more than one way how the keywords of the original query can be arranged in order to obtain a valid query in the Base Language. Moreover, there is rarely only one way of possible arrangements. It is, therefore, necessary to develop a measure of the query translation results in order to range them by the probability that this particular result is actually the one the user has intended to ask. To create such a measure, the concepts of entropy and penalties were introduced in the system.

VI. CONCEPTS OF ENTROPY AND PENALTY

The concept of entropy (meaning – the distance between the original query and its translation) was introduced in the query translation process in order to establish a metric for evaluating the translation results. The original query entered by the user can be understood in different ways by the system, and for each of those translation results, the entropy is calculated. The entropy value zero means that there have been made no changes in the original query (i.e. the original query is itself a valid query in the Base Language). The more changes that need to be made in the query in order to get a valid query in the Base Language there are, the bigger the entropy gets. The calculation of the entropy is done by assigning the so-called penalties for various types of transformations (called the primitive operations) than can be done with the original query in order to turn it into a valid query in the Base Language. The types of penalties and the process of the construction of the entropy are described in more detail in [18].

The main idea behind the concept of entropy is the ability to range the translation results for the query in increasing order by their plausibility and then show the most plausible ones back to the user. Since the Base Language proved to be very well readable by the domain experts [1], we firmly believe that the experts should be able to understand the provided translation results and pick the one that corresponds to the idea that was in their minds during the query formulation phase. Then the system can take into account the query chosen by the user and configure the penalties applied to that particular translation result so that next time such type of translation would get smaller entropy (i.e. it will have a bigger probability of being the correct one).

VII. CONFIGURATION OF PENALTIES

The configuration of penalties is the feature that is built in the system for more advanced end-users that are willing to increase the probability of the query translation process. Typically, the probability of the process increases on its own over time while the user exploits the system. However, if the user is aware of his/her own specifics, it is possible to intervene in the system's built-in behavior.

To increase the probability of the query translation, one must achieve a decline in the overall entropy of the

preferred type of query translation results. To do this one must first understand the basics of the entropy calculation [18] and secondly alter the original penalties assigned to various kinds of primitive operations.

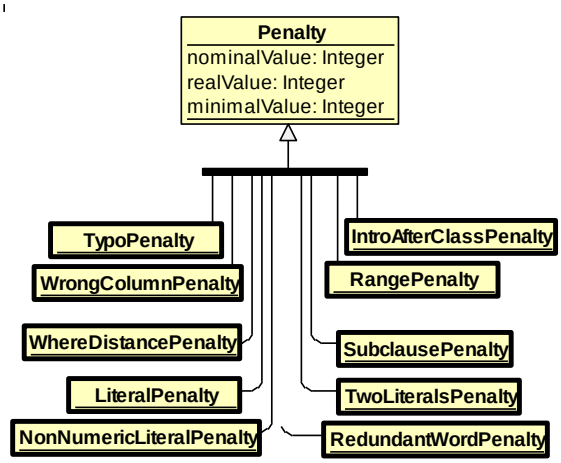

Figure 2. The penalty metamodel.

To alter the abovementioned penalties, one must understand a bit more about the way the system treats the penalties. Fig. 2 shows the hierarchy of the penalties described in more detail in [18] together with their most important attributes that can be configured. The system provides a means for altering the values of those attributes.

Each subclass of the superclass "Penalty" is a singleton class as can be seen in Fig. 2 (denoted by the bold frame) which means they all have only one instance each. The three attributes of each of those instances (defined in the superclass "Penalty") denotes the starting values for the entropy calculation mechanism. The attribute "nominalValue" denotes the nominal starting penalty, the attribute "realValue" denotes the current penalty, and the attribute "minimalValue" denotes the minimal allowed penalty for this type of the primitive operations. User can intervene in the entropy calculation process by altering the values of these attributes.

VIII. Conclusion

This paper describes the work in progress. It has to be read as a continuation of the work described in [18], [19]. We have developed a prototype implementing the keywords-containing test-based querying. We have as well implemented the calculation of the entropy of the results of the query translation. Also, we have performed some laboratory experiments with user experience-based learning. The approach, in general, still needs to be verified in some real-world use-cases.

The system described in this paper still has the potential for improvements. One example of such improvements is to take into account not only the underlying data ontology but also the data items themselves. It should improve the accuracy of the query translation process because some of the words of the original query that are currently unclear for the translator could be identified as existing attributes of the underlying real data.

Another research direction that is not yet inspected is to involve the machine-learning in the process of query translation.

Conflict of Interest

The author declares no conflict of interest.

Acknowledgment

This work is supported by the ERDF PostDoc Latvia project Nr. 1.1.1.2/16/I/001 under agreement Nr. 1.1.1.2/VIAA/1/16/218 "User Experience-Based Generation of Ad-hoc Queries From Arbitrary Keywords-Containing Text".

References

[1] J. Barzdins, M. Grasmanis, E. Rencis, A. Sostaks, and J. Barzdins, "Ad-Hoc querying of semistar data ontologies using controlled natural language," *Frontiers in Artificial Intelligence and Applications*, vol. 291, pp. 3-16, 2016.

[2] E. Rencis, J. Barzdins, M. Grasmanis, and A. Sostaks, "Facilitation of health professionals responsible autonomy with easy-to-use hospital data querying language," in *Proc. the 13th International Baltic Conference on Databases and Information Systems*, 2018, pp. 1-14.

[3] D. D. Chamberlin and R. F. Boyce, "SEQUEL: A structured English query language," in *Proc. ACM SIGFIDET Workshop*, Ann Arbor, Mich., May 1974, pp. 249-264.

[4] E. Prud'hommeaux and A. Seaborne. (January 2008). SPARQL query language for RDF. W3C Recommendation. [Online]. Available: http://www.w3.org/TR/rdfsparql-query

[5] M. Zviedris and G. Barzdins, "ViziQuer: A tool to explore and query SPARQL endpoints," *The Semantic Web: Research and Applications*, vol. 6644, pp. 441-445, 2011.

[6] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch, "Natural language interfaces to databases – An introduction," *Natural Language Engineering*, vol. 1, no. 1, pp. 29-81, 1995.

[7] F. Li and H. V. Jagadish, "Constructing an interactive natural language interface for relational databases," *Journal Proceedings of the VLDB Endowment*, vol. 8, no. 1, pp. 73-84, 2014.

[8] 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.

[9] N. Papadakis, P. Kefalas, and M. Stilianakakis, "A tool for access to relational databases in natural language," *Expert Systems with Applications*, vol. 38, pp. 7894-7900, 2011.

[10] A. M. Popescu, A. Armanasu, O. Etzioni, D. Ko, and A. Yates, "Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability," in *Proc. the 20th International Conference on Computational Linguistics*, 2004, article no. 141.

[11] L. Fei and H. V. Jagadish, "NaLIR: An interactive natural language interface for querying relational databases," in *Proc. the ACM SIGMOD International Conference on Management of Data*, 2014.

[12] T. Gao, M. Dontcheva, E. Adar, Z. Liu, and K. G. Karahalios, "DataTone: Managing ambiguity in natural language interfaces for data visualization," in *Proc. the 28th Annual ACM Symposium on User Interface Software & Technology*, 2015, pp. 489-500.

[13] J. Barzdins, E. Rencis, and A. Sostaks, "Granular ontologies and graphical in-place querying," *Short Paper Proceedings of the PoEM*, vol. 1023, pp. 136-145, 2013.

[14] J. Barzdins, E. Rencis, and A. Sostaks, "Data ontologies and ad hoc queries: A case study," in *Proc. the 11th International Baltic Conference*, 2014, pp. 55-66.

[15] E. Rencis, "On keyword-based ad-hoc querying of hospital data stored in semistar data ontologies," *Procedia Computer Science Journal*, vol. 138, pp. 27-32, 2018.
[16] E. Rencis, "Towards a natural language-based interface for querying hospital data," in *Proc. International Conference on Big Data Technologies*, Hangzhou, China, 2018, pp. 25-28.
[17] E. Rencis, "Natural language-based knowledge extraction in healthcare domain," in *Proc. the 3rd International Conference on Information System and Data Mining*, Houston, Texas, USA, 2019, pp. 138-142.
[18] E. Rencis, "User experience-based information retrieval from semistar data ontologies," in *Proc. the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, 2019, vol. 1, pp. 419-426.
[19] E. Rencis, "Knowledge extraction from healthcare data using user-adaptable keywords-based query language," in *Proc. the 4th International Conference on Information System and Data Mining*, Hilo, Hawaii, USA, 2020, in press.

Edgars Rencis was born in Cesis, Latvia on October 20, 1982. He earned his PhD in Computer Science in the University of Latvia, Riga, Latvia in 2012 with specialization in programming languages and tools.

He is currently working as the leading researcher for the Institute of Mathematics and Computer Science, University of Latvia, Riga, Latvia. He is also an assistant to professor in the Faculty of Computing, University of Latvia. He has been one of the leading researchers in several ERDF and Latvian National research projects. He has participated in the team of scientists working to develop the first version of a natural language-based ad-hoc querying tool for a national hospital of Latvia. His main areas of research include modelling and metamodelling, model transformation languages and tools, ontologies and domain-specific languages in the domain of medical management.