APANTISIS: A Greek Question-Answering System for Knowledge-Base Exploration

Conference Paper · September 2016

CITATIONS
0

READS
79

3 authors:

Emmanouil Marakakis
Technological Educational Institute of Crete
21 PUBLICATIONS **93** CITATIONS
SEE PROFILE

Haridimos Kondylakis
Foundation for Research and Technology - Hellas
83 PUBLICATIONS **492** CITATIONS
SEE PROFILE

Aris Papakonstantinou
Technological Educational Institute of Crete
4 PUBLICATIONS **1** CITATION
SEE PROFILE

APANTISIS: A Greek Question-Answering System for Knowledge-Base Exploration[1]

Emmanouil Marakakis[1, a)], Haridimos Kondylakis[2, b)], Papakonstantinou Aris[1, c)]

[1]*Department of Informatics Engineering, Technological Educational Institute of Crete*
Heraklion, GR-71410, Greece
[2]*Computational Biomedicine Laboratory, FORTH-ICS, N. Plastira 100, Heraklion, GR70013, Greece*

[a)]mmarak@cs.teicrete.gr
[b)] Corresponding author: kondylak@ics.forth.gr
[c)] aris.papakonstantinou@outlook.com

Abstract. As users struggle to navigate on the vast amount of information now available, methods and tools for enabling the quick exploration of the databases content is of paramount importance. To this direction we present *Apantisis*, a novel question answering system implemented for the Greek language ready to be attached to any external database/knowledge-base. An ingestion module enables the semi/automatic construction of the data dictionary that is used for question answering whereas the Greek Language Dictionary, the Syntactic and the Semantic Rules are also stored in an internal, extensible knowledge base. After the ingestion phase, the system is accepting questions in natural language, and automatically constructs the corresponding relational algebra query to be further evaluated by the external database. The results are then formulated as free text and returned to the user. We highlight the unique features of our system with respect to the Greek language and we present its implementation and a preliminary evaluation. Finally, we argue that our solution is flexible and modular and can be used for improving the usability of traditional database systems.

INTRODUCTION

The exponential growth of the web and the development of new scientific techniques have led to an information explosion. According to the 2011 Gartner Group Report[2], the worldwide information volume is growing annually at a minimum rate of 59%. A prime obstacle for non-technical people who wish to exploit the aforementioned information has been the need to either learn a special language for communicating with the machine or communicate via an intermediary.

To this direction, there is an ever growing interest in allowing users to ask questions using natural language in the hope of improving the usability of information systems. This need has already been recognized by many commercial vendors such as Microsoft, Apple, Amazon and Google which are currently providing prominent question answering systems. Microsoft Cortana, Apple Ciri, Amazon Echo and Google Now are being rapidly developed and updated daily showing the renewed increasing interest on efficient and effective question answering systems.

In this paper we use the notions of *database* and *knowledge-base* interchangeably referring to systems adopting the entity relationship (E-R) model [1], assuming that our data conform to the aforementioned model. The reason for this assumption is that there is a well-defined mapping between a natural language sentence in English and the E-R model [2]. We considered and applied the corresponding mapping for the Greek language and the E-R model as

[1] This work was partially support by the iManageCancer (H2020-643529) EU project.
[2] http://web.archive.org/web/20110710043533/http://www.gartner.com/it/page.jsp?id=1731916

well. The benefit of this approach is that users of a database can make their queries directly in the Greek language rather than in formal languages like SQL. The system *Apantisis* automatically transforms the Greek question into the corresponding Prolog query for retrieving the requested data. The answer is also given in Greek rather than in the form of tuples. More specifically the contributions of this paper are the following:

- We present Apantisis, a novel platform that accepts a user's natural Greek language input and provides explicit answers to questions. The system is composed of three main modules, the *question*, the *answer* and the *ingestion* modules.
- The *question* module accepts user's question and converts it into an equivalent relational algebra expression. The query is then forwarded to the database to be answered [3].
- The *answer* module accepts user's question and the results from the database and formulates the appropriate answer which is presented to the user [4].
- To enable effective question answering a data dictionary should be available at setup time for the external database. This data dictionary can be manually created and provided by the users or automatically extracted using the *ingestion* module.

Our system is unique in the sense that it exploits the unique characteristics of the Greek language to enable rapid and effective database exploration for the users not possessing technical skills.

The remaining of this paper is structured as follows: A running example is shown in Section 2. Then the architecture of our system and the different components are described in Section 3. Related work is reported in Section 4. Finally, Section 5 concludes this paper and presents a preliminary evaluation.

RUNNING EXAMPLE

The most common questions to a QA system can be separated in three basic categories:

- Factual or interrogative Questions: These questions have only one correct answer. For example, "`Which lessons does professor Marakakis teach?`".
- Opinion or declarative Questions: These questions ask for some kind of opinion, belief or point of view, so they have yes or no answer. For example, "`Professor Marakakis teaches Artificial Intelligence`". This declarative statement may be correct or wrong .
- Summary or imperative Questions. These questions have more than one answer, but they still must be supported with evidence. For example, "`List all lecturers of Informatics Engineering Department for academic year 2015-16.`".

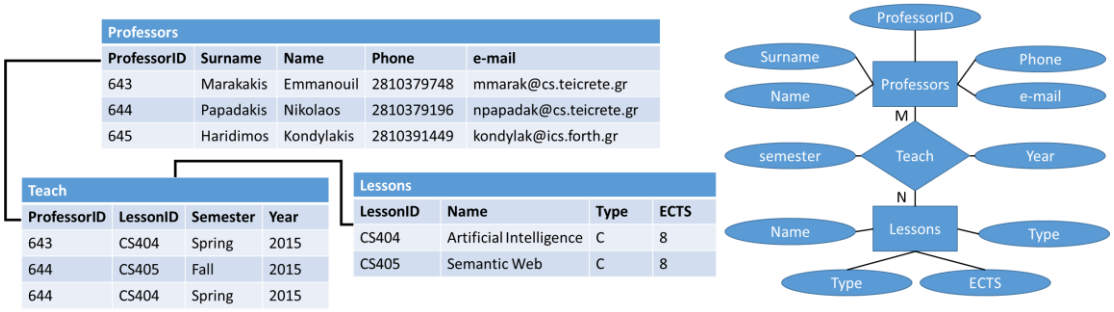

FIGURE 1. An example database (right) and the corresponding E-R model (left).

In this work we focus only on these three basic categories of questions. Consider for example the toy schema shown in Figure 1 containing professors and the lessons they teach. In order to identify the courses taught by professor Marakakis an experienced user should know the schema and he should formulate the following query - expressed in SQL:

```
SELECT Lessons.Name FROM Lessons, Teach, Professors
WHERE Lessons.LessonID=Teach.LessonID AND Teach.ProfessorID=Professors.ProfessorID
  AND Professors.surname="Marakakis"
```

On the other hand, it would be a lot easier for an unexperienced user just to use the following natural language query: "`Which lessons does professor Marakakis teach?`"

ARCHITECTURE

APANTISIS is composed of six individual modules which can be attached to any existing database enabling natural language query answering using the Greek language. The six modules are shown in Figure 2 and they are a) *the GUI*, b) *the question*, c) *the answer*, d) *the ingestion*, the e) *Greek Language Lexicon, Syntactic and Semantic rules* and f) *the data dictionary*. Bellow we will describe in detail each one of the aforementioned modules.

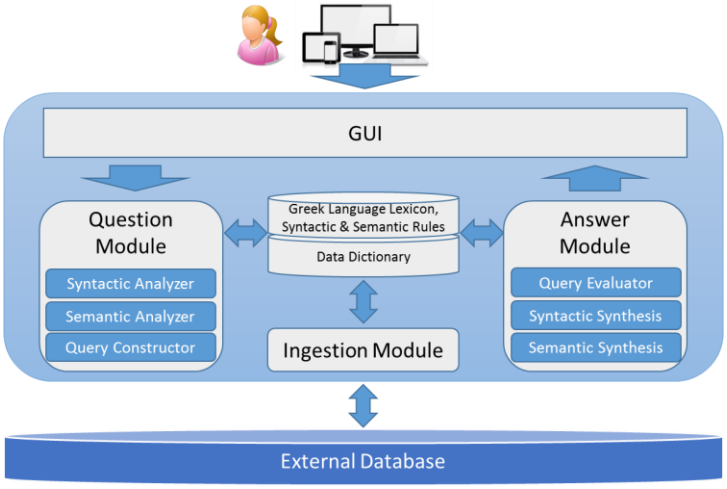

FIGURE 2. The high level architecture of our system.

Professor	
ProfessorID	Integer
Surname	Nvarchar(20)
Name	Nvarchar(20)
Phone	Nvarchar(20)
e-mail	Nvarchar(20)
Teaches	
ProfessorID	Integer
LessonID	Integer
Semester	Nvarchar(20)
Year	Integer(4)
Lesson	
LessonID	Integer
Name	Nvarchar(50)
Type	Nvarchar(1)
ECTS	Integer(2)

FIGURE 3. The extracted meta-data.

THE INGESTION MODULE

Before actually using the system the necessary data dictionary should be in place. The ingestion module connects to the external database and retrieves all relevant meta-data required for constructing the data dictionary. This data dictionary records various meta-data about the database such as the table and the column names and types and the foreign key constraints. In our example of Figure 2 the names of the tables, the names and types of their corresponding fields and the foreign key constraints are extracted and saved at the data dictionary as shown in Figure 3.In case that the database tables and fields use abbreviations, the administrator is allowed to assign proper, meaningful names to those by annotating the corresponding fields. For example, for the field "ECTS", the administrator can add the annotation "European Credit Transfer and Accumulation System". Finally, synonyms can be added to the identified terms as well. For example, in our example the administrator can indicate in the data dictionary that ''lecturer" is synonym to "teacher".

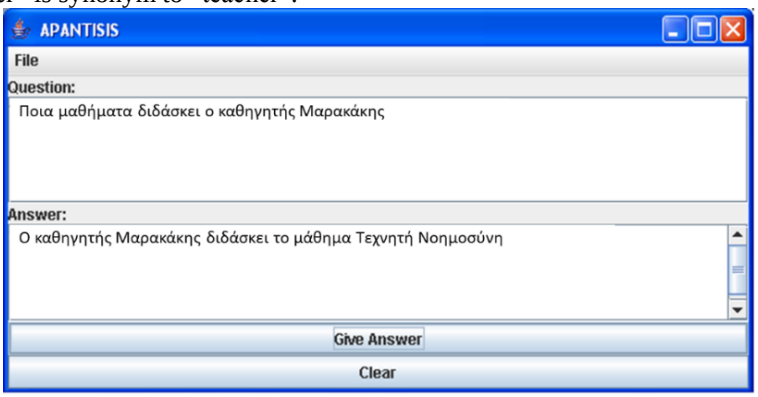

FIGURE 4. The GUI of the question answering prototype. The query corresponds to the sentence "Which lessons does professor Marakakis teach?" and the answer corresponds to the "The professor Marakakis teaches the lesson of Artificial Intelligence".

THE GUI

The GUI is shown in Figure 4 and is developed using JAVA. The Jasper library is used to communicate with the rest of the system which is developed in Prolog. As shown, a user can specify a question using the corresponding textarea field and submit his question. The system transforms the natural language question to the corresponding query using the question module and retrieves the results from the database. Finally, the results are transformed to a natural language sentence by the answer module and they are shown to the user. In our running example, the user issues the following question "*Which lessons does professor Marakakis teach?*" shown in Greek language in Fig. 4.

THE QUERY MODULE

As soon as the question of the user is received, the syntactic analyser parses the query and constructs the parse tree of each sentence. To do that, initially, the syntactic analyser exploits *the Greek language dictionary* as shown in Figure 2. In this step tokenization, stemming and lemmatization is performed in order to reduce inflectional and derivationally related forms of a word to a common base form.

Tokenization is the recognition of word and punctuation boundaries inside the text. This involves an initial split at obvious points (spaces and punctuation marks) followed by post processing to avoid for example the separation of the relative indefinite pronoun "*ό,τι*" into two different tokens. Then stemming and lemmatization is following. Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. Lemmatization on the other hand, refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. If confronted with the token "*saw*", stemming might return just "*s*", whereas lemmatization would attempt to return either "*see*" or "*saw*" depending on whether the use of the token was as a verb or a noun.

The Greek language, like many others, uses extensive inflection and therefore semantic relations between words cannot be detected unless those words are stemmed. Stemming Greek words is much harder than stemming words of other European languages because of the number of possible suffixes, many of which cannot be properly separated from the word stem without knowing the grammatical type of the word. Moreover, stemming of words with different meaning may lead to the same stem. In addition, although for the English language there are already established many approaches and heuristics with respect to stemming and lemmatization, for the Greek language we had to design and implement all of them from scratch. Then the syntactic analyzer constructs the parse tree of the sentence exploiting *the grammar rules of the Greek language* and then the semantic analyser identifies the elements of the query that correspond of the data dictionary and ascertains the correct usage of each. Synonyms can be identified and the correspondences between the user's query and the names of the tables and fields are matched. In our running example, the parse tree shown in Figure 5(a) will be constructed.

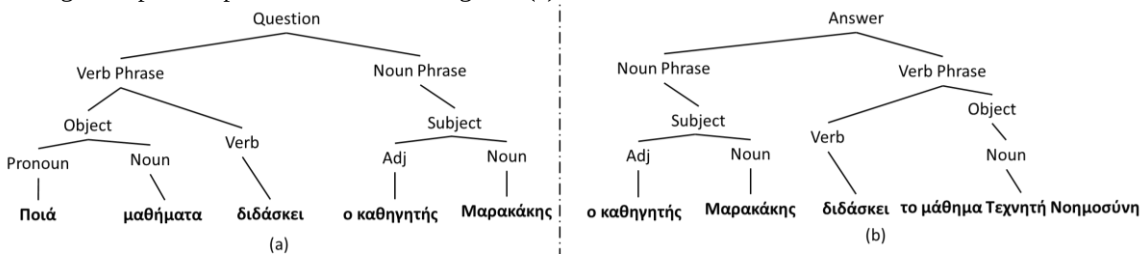

FIGURE 5. The parse tree of the question (a) and the corresponding answer (b).

After the syntactic and the semantic analysis of the question of user, the *query constructor* generates the corresponding query in relational algebra. The steps for creating the corresponding query are the following: *a) Identify Tables*: To construct the corresponding query initially the query module tries to identify the tables involved in the query. In our running example, the words of the query are grammatically classified in the parse tree and then the data dictionary is used to match with specific DB elements. In our example, the words "καθηγητής", "διδάσκει" and "μαθήματα" of the analyzed question correspond to the database relations "*Professor", "Teaches"* and *"Lessons"* respectively. Note that in this stage we exploit the entity relationship model in the sense that we try to correlate verbs with relationships and nouns with entities. *b) Construct Join Path*: Next we try to identify the paths joining those tables using the foreign key constraints identified in the ingestion phase. If there are multiple paths joining the identified tables, the one with the minimum length is used. In our case for example the "*Professor"* table

is joined with the *"Teaches*" table using the *"ProfessorID"* field, and the *"Lesson"* table is joined with the "*Teaches*" table using the *"LessonID"* field. To identify the proper join path our system exploits the entity relationship model. *c) Identify Selections and Projection*: Then the query module tries to identify the projection part of the query exploiting the Greek Language Syntax and Semantic rules. In our case the projection will be on the field "*Name*" of the lesson table. Finally, it will try to identify the selections of the query. In order to do that it searches all field of the table professor to identify records with the "*Marakakis*" value. Of course since we know the datatypes of the fields we are looking only at the fields with the correct datatypes - (Nvarchar in our case). Since the specific value is the "*Surname*" field the corresponding condition is added to the query.

The query can be directly translated into the corresponding SQL/relational algebra query and sent to the database. In our case it is translated to the following Prolog query: `Q(L):-Professors(P, Marakakis, N, P, E), Teach(P, L, S, Y), Lessons(L, Y, T, C)`

THE ANSWER MODULE

This module is responsible for executing the query and transforming the results into a natural language expression. As such, the Answer module receives the parse tree of the question and the query as input. The corresponding algorithm proceeds as follows: Initially the Query Evaluator component of the Answer module sends the constructed query to the external database and collects the results. In our case the Prolog query returns the "*Artificial Intelligence*". The second step of the corresponding algorithm constructs the parse tree of the answer using the parse tree of the question, the results of the knowledge base, the Greek language lexicon and the data dictionary. The noun phrase and the verb of the question are transferred directly to the noun phrase and the verb of the answer as shown in Figure 5(b). The remaining answer is constructed using the query results and the various phrase structure rules of the Greek language. Finally, in the last step of the corresponding algorithm the parse tree of the answer is checked and creates the final answer. In our running example the answer would be "*Professor Marakakis teaches the lesson of Artificial Intelligence*".

RELATED WORK

Work on question answering systems as front-ends to databases has been going on since the 1970s, however the systems there where limited only to a narrow domain. More recent works include systems that use NLP for accessing databases using the E-R model [2] or try to transform a natural language question into a corresponding logical form by leveraging semantic associations between lexical representation and database properties in the latent space [6]. In the same direction Badia [7] introduces a query language with generalized quantifiers to bridge the gap between natural language queries and database querying. However, despite the fact that questions are transformed to a logic form equivalent to a relational query language they do not offer natural language answers as well. In addition, there is a strong community behind question answering systems using the English language with several tracks on well-respected conferences. For example, the CLEF question answering track [8] benchmarks related systems allowing access over information using natural language questions. However, there is no corresponding Greek language benchmark to evaluate our system.

For the Greek language there already have been some preliminary approaches on Natural Language Processing. For example, the Institute for Language and Speech Processing have created a set of tools [5], [9] based on both machine learning and rule-based algorithms for natural language processing tools for Greek. However, the aforementioned tools do not support query answering and are used mainly for whole text summarization. Adam et al. [10] implement stemming and tagging for the Greek language based on word tagging techniques whereas Ntais in [11] uses the Porter Stemmer [12] to achieve stemming for the Greek language. Approaches like [13] have created a large greek-english dictionary with incorporated speech and language processing tools implementing lemmatizer for Greek and Text to Speech synthesizers for Greek. Although these tools contribute to the stemming and tagging of the Greek language still they only cover specific sub-modules of our system.

CONCLUSION AND PRELIMINARY EVALUATION

To our knowledge, our approach is the only approach combining both question and answering in natural language in the Greek language. The system enables the specification of natural language questions, finds the

corresponding answers and presents them using natural language as well. The combination of natural language processing for both the question and answer part and the Greek language implementation make the system unique. Its generality and modularity along with the ingestion module allow the quick deployments of the system on various databases with minor effort.

To evaluate our system, since there is no question answering evaluation benchmark available for the Greek language, we conducted a preliminary evaluation using available course data from the department of Informatics Engineering at the Technological Educational Institute of Crete. The benchmark is consisted of 14 tables published with the corresponding data from the department and 11 relevant query templates corresponding to hundreds of similar queries. The generated benchmark can be found online[3] along with the results from our system allowing other future systems to be evaluated using the aforementioned benchmark. All queries were successfully answered returning the expected results from the system. In addition, we tried to issue questions where the syntax was not correct and our system was able to identify the erroneous syntax and to notify the user. In the case that we are searching for tuples where there is no corresponding information in the database the system was able to correctly identify that there is no relevant information in the database.

As future work, we intend to conduct a thorough evaluation and go beyond these three simple types of questions. That is questions can be more complex to include multiple anaphoras. For example, "Which course is taught by professor Marakakis which has more than 100 students and which is given in Spring semester?". In addition, we would like to support more complex queries like why-not types of queries. Moreover, we would like to focus on correcting syntax and gramma errors of the input questions. Finally, building a combined portfolio of structured, semi-structured and unstructured knowledge bases is an obvious extension we already started to explore [14], [15].

REFERENCES

1. P.Chen, The Entity-Relationship Model-Toward a Unified View of Data, ACM TODS, 1(1):9-36 (1976).
2. P. Chen, "English Sentence Structure and Entity-Relationship Diagram". IS Journal, 1(1):127-149, (1983).
3. K. Varouxis, "Development of a System which Analyses Queries in Greek Language and Transforms them into Prolog Goals", Department of Applied Informatics and Multimedia, TEI of Crete (2007).
4. G. Antalis, "Development of a System which Generates Sentences in Greek Language as Answers from Processing Queries in a Relational Database", Department of Applied Informatics and Multimedia, TEI of Crete (2008).
5. H. Papageorgiou, P. Prokopidis, I. Demiros, V. Giouli, A. Konstantinidis and S. Piperidis, "Multi-level XMLbased Corpus Annotation". Language Resources and Evaluation Conference. Las Palmas (2002).
6. M.C Yang, N. Duan, M. Zhou, H.C. Rim, "Joint Relational Embeddings for Knowledge-based Question Answering," EMNLP, 645-650 (2014).
7. A. Badia, "Question answering and database querying: Bridging the gap with generalized quantification," J. Applied Logic 5(1): 3-19 (2007).
8. A. Peñas, C. Unger, G. Paliouras, I.A. Kakadiaris, "Overview of the CLEF," QA Track. 539-544 (2015).
9. P. Prokopidis, B. Georgantopoulos, H. Papageorgiou, "A suite of Natural Language Processing tools for greek," The 10th International Conference of Greek Linguistics (2011).
10. G. Adam, K. Asimakis, C. Bouras, and V. Poulopoulos, "An Efficient Mechanism for Stemming and Tagging: The Case of Greek Language", KES 3, 389-397 (2010).
11. G. Ntais, "Development of a stemmer for the Greek language," MSc Thesis, Stockholm University (2006).
12. M.F. Porter, "An algorithm for suffix stripping," Program; automated library and information systems 14(3): 130–137 (1980).
13. D.P. Lyras, G.K Kokkinakis, A. Lazaridis, K.N. Sgarbas, N. Fakotakis, "A large greek-English dictionary with incorporated speech and language processing tools". INTERSPEECH, 1891-1894 (2009).
14. H. Kondylakis, L. Koumakis, E. Kazantzaki, M. Chatzimina, M. Psaraki, K. Marias, M. Tsiknakis, "Patient Empowerment through Personal Medical Recommendations", MEDINFO, 216, 1117 (2015).
15. H. Kondylakis, L. Koumakis, M. Psaraki, G. Troullinou, M. Chatzimina, E. Kazantzaki, K. Marias, M. Tsiknakis, "Semantically-enabled Personal Medical Information Recommender", ISWC (2015).

[3] http://users.ics.forth.gr/~kondylak/Benchmark.rar