A Medical Question Answering System with NLP and graph database

Ioannis Tsampos [1], Emmanouil Marakakis [1]

[1]*Department of Electrical & Computer Engineering, Hellenic Mediterranean University, Heraklion, Crete, 71410, Greece*

Abstract

A Question Answering (QA) System has been developed that is able to process medical and other documents in Greek language and to provide answers to user questions. It integrates Natural Language Processing (NLP) methods and a graph database. It retrieves information from text and creates an efficiently searchable graph using rule-based matching methods. Search results are provided to the user as an answer to question in Greek language. Our approach is natural language independent and domain independent. Moreover, our approach handles efficiently complex queries and large volumes of texts. In medicine, our approach can be used for smart healthcare applications which require QA support.

Keywords

Question Answering, Natural Language Processing, Medical Knowledge, Greek Language, Graph Databases, Query languages

1. Introduction

There is a growing trend for users to ask for specific answers to their questions rather than lists of search results. A typical example is the chatGTP[1] where, as soon as it came available to the public, a large number of users rushed to try it asking questions of any difficulty. Many users adopted it in everyday tasks. It is important for average users to be able to receive fast, clear and accurate answers to their questions especially with regard to their health.

Having in mind that it is important for everyone to be informed accurately about medical matters, we developed a medical question answering system. The system is intended to retrieve information from professional documents and to provide it to the average user as answers to questions.

We have designed and developed a system that analyzes texts and represents the parse tree as nodes and relationships of a graph database. The relationships are enriched analyzing grammatical and logical dependencies between nodes. The graph data can be searched using graph database queries with high retrieval efficiency. The system processes user questions and converts them to database queries. The results are returned to the user in the natural language as a Greek text.

The scope of our research is to propose an implementation of a highly-efficient QA system utilizing modern technologies and open-source tools. Additionally, we want to study the efficiency and accuracy of such a system in the medical domain.

2. Related work

Over the last years research on automated question answering systems has rapidly increased. It is indicative that the count of QA publications in Association for Computational Linguistics

[1] https://openai.com/blog/chatgpt/

Published in the Workshop Proceedings of the EDBT/ICDT 2023 Joint Conference (March 28-March 31, 2023), Ioannina, Greece
EMAIL: tsamposg@uoc.gr (I. Tsampos); mmarak@hmu.gr (E. Marakakis)
ORCID: 0009-0000-1446-9377 (I. Tsampos); 0000-0002-5685-0480 (E. Marakakis)

(ACL) had more than doubled between years 2017 and 2020 [1]. There are many types of QA systems according to the classification presented in the aforementioned survey. According to the modeling approach there are Rule Based, Machine Learning and Deep Learning Systems.

On medical QA Systems domain, research has been done on rule based [2] and machine learning [3] implementations.

In Greek language, the research on NLP and QA systems compared to other languages is limited. According to the paper "NLP for the Greek Language: A Brief Survey" [4], published in 2020 the only publication about QA in the Greek language was the "APANTISIS, a modular QA system implemented for the Greek language for plugging it to databases or knowledge bases." [5]. The system accepts questions to Greek language and returns answers based on database tables.

Since then, there were 2 more papers published on Question Answering in Greek language. The first one is an intelligent chatbot which relies on semantic web technologies and offers an intelligent controlled natural language interface for accessing the information available in DBpedia [6]. The second is a research prototype named Tiresias that relies on Machine Translation tools, and BERT QA models to implement Bilingual Question Answering over DBpedia Abstracts [7].

To the best of our knowledge no other implementation is currently available for Question Answering on medical or other domain in the Greek language able to process free-text documents.

On the other hand, there are many implementations of Medical [2] and Open Domain Question [8] Answering Systems in English [9] and other languages, based on machine learning [3] and rule-based methods. A common approach is the question answering over knowledge graph, which aims to use facts in knowledge graph to answer the questions [10]. Another approach is the Grammatical Question Answering [11] for parsing questions posed in natural language by means of Grammatical Framework and then transforming them to SPARQL queries.

3. Approach

The application has been developed in Python programming language, using the open-source natural language processing library spaCy, the graph database management system Neo4j and the graph query language Cypher. We also used the Neo4j graphical interface for visualization of the graph.

We use the Neo4j[2] graph database to store the parse tree extracted by spaCy[3]. Tokens extracted by spaCy POS tagger are transferred to the database as nodes. Grammar relationships extracted by spaCy Dependency Parser are transferred as relationships between nodes (Figure 1). We use Cypher to query the graph database according to the user's question and we provide the answer based on the returned results (Figure 2).

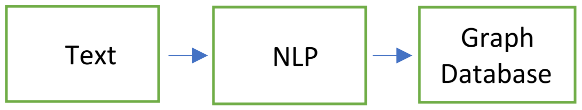

Figure 1: Data Storage Process

3.1. spaCy NLP

spaCy is an open-source natural language processing library for Python that can be used to tokenize, tag, and parse natural language text. It provides tools for NLP processing and pre-trained models, in many languages including Greek [12] and English. Among other tools provided by SpaCy, the POS tagger categorizes words as Parts of Speech, the Morphologizer assigns morphological features and coarse-grained POS tags following the Universal Dependencies UPOS and FEATS annotation guidelines and the Dependency Parser creates and describes syntactic functions of distinct words in a phrase.

We used spaCy for Dependency Parsing assigning word types to tokens describing the grammar relationships between them.

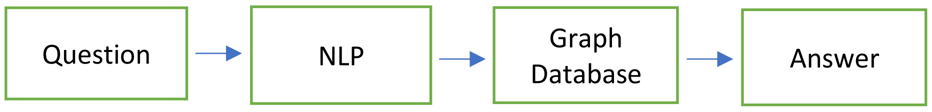

Figure 2: Question Answering Architecture

[2] https://neo4j.com/

[3] https://spacy.io/

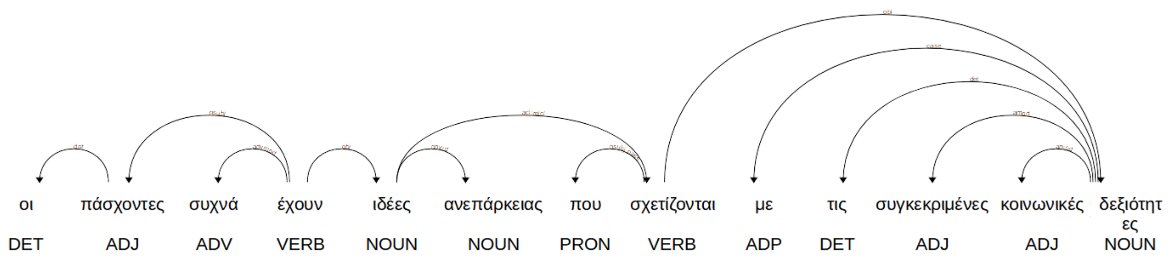

Figure 3: Parse tree created by spaCy

3.2. Neo4j graph database and Cypher query language

Neo4j is a graph database that is well-suited for storing and querying complex relationships between data [13]. A graph data structure consists of nodes (discrete objects) that can be connected by relationships. Nodes are the entities in the graph that can be tagged with labels, representing their different roles. Nodes can hold any number of key-value pairs, or properties. Node labels may also attach metadata to certain nodes. Relationships describe a connection between a source node and a target node and provide directed, named, connections between two node entities. Relationships always have a direction, a type, a start node, and an end node, and they can have properties, just like nodes. Relationships can be navigated in any direction. Graph databases are storing highly variable data that is difficult to contain in a pre-defined schema. Because graph databases focus on the relationships between instances, they are a natural choice to store relationally focused data [14].

Cypher [15] is a declarative query language for Neo4j optimized for graphs. Cypher allows efficient data storage and retrieval from the graph. Cypher's expressive syntax allows modeling the relationships between data in a way that is intuitive and easy to understand.

3.3. Implementation

The application reads a text, creates the parse tree, and represents it as a graph in the graph database. The processes of making a parse tree includes the following steps:

1. *Sentence segmentation*: the text is divided into sentences and each one is processed separately.
2. *Tokenization*: The text is segmented into chunks of information that can be considered as discrete elements like words, numbers, symbols, and punctuation.
3. *Part-of-speech (POS) tagging & morphological analysis*: Each word is grammatically classified after morphological analysis. A word is classified as VERB, NOUN, etc and its morphological features like VerbForm, Mood, Gender, Case, etc are specified.
4. *Dependency Parsing*: Assignment of syntactic dependency labels, describing the relationships between individual tokens, like subject or object.

spaCy uses statistical models for parsing and tagging and makes predictions of which tag or label most likely applies in this context.

As an example, we have the sentence "*Sufferers often have ideas of inadequacy related to specific social skills.*", and the Greek translation "*Οι πάσχοντες συχνά έχουν ιδέες ανεπάρκειας που σχετίζονται με τις συγκεκριμένες κοινωνικές δεξιότητες*". After the sentence is processed, all the tokens are labeled as relevant to a particular part of speech. Also, dependency labels between words are added, describing their grammatical relationship. The data extracted after the POS tagging and morphological analysis are shown in Table 1 (only verbs, nouns and adjectives are included). The parse tree of the Greek sentence is shown in Figure 3.

Parse tree is transferred to the graph database. The database's nodes and relationships are created with Cypher queries according to the following rules:

1. Each token is represented by a Node which has the token's name. Each Node

is labeled by the name of POS tag of the token.
2. Each dependency is represented as a relationship between two Nodes.
3. Word text, lemma, token position in the sentence and all the morphological features for each token are stored as properties of the Nodes.

The text given by a user for information extraction, is split into sentences. Each sentence is processed, and the related data are stored in the graph database. After the completion of the processing of our example sentence, the data can be visualized as shown in Figure 4.

Table 1
POS Tagging and Morphological analysis data

TOKEN	POS	MORPHO-LOGICAL FEATURES
πάσχοντες	ADJ	Case=Nom, Gender=Masc, Number=Plur
έχουν	VERB	Aspect=Imp, Mood=Ind, Number=Plur, Person=3, Tense=Pres, VerbForm=Fin, Voice=Act
συχνά	ADV	-
ιδέες	NOUN	Case=Acc, Gender=Fem, Number=Plur
ανεπάρκειας	NOUN	Case=Gen, Gender=Fem, Number=Sing
σχετίζονται	VERB	Aspect=Imp, Mood=Ind, Number=Plur, Person=3, Tense=Pres, VerbForm=Fin, Voice=Pass
συγκεκριμένες	ADJ	Case=Acc, Gender=Fem, Number=Plur
κοινωνικές	ADJ	Case=Acc, Gender=Fem, Number=Plur

The size and the color of each node are related to its label. The nodes labeled as "NOUN" are blue, the nodes labeled as "VERB" are red, the nodes labeled as "ADJ" are orange and the node labeled as "ADV" is green. Other node labels like "DET" and "PRON" are hidden for simplicity.

Information in database can be retrieved using Cypher queries. The syntax is quite simple. We construct a query to get the subject of the phrase, so we need the token that is connected to the verb "έχω" ("have") with the relationship nsubj. The query is:

```
match (who)-[:nsubj]-(VERB{name:'έχω'})
return who
```

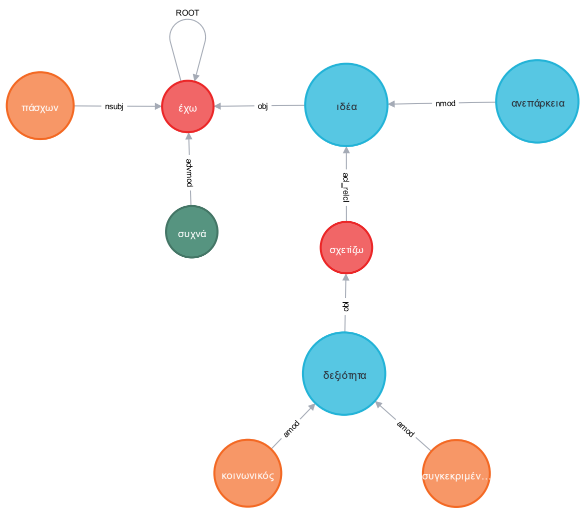

Figure 4: The parse tree stored in graph database as Nodes and Relationships

The query returns as result the node named "πάσχων" (sufferer). If it is necessary, we can retrieve the initial text "πάσχοντες" ("sufferers") which is included in the node's properties.

The query can be more specific by including the object noun "ιδέα" ("idea")

```
match p=(who)-[r1:nsubj]-(VERB{name:'έχω'})-
[r2:obj]-(NOUN{name:'ιδέα'}) return p
```

The result will be the same as the previous query. Another scenario is to ask for the object of a phrase. In our example the query will be:

```
match ({name:'πάσχων'})-[:nsubj]-
(VERB{name:'έχω'})-[:obj]-(what) return what
```

The query returns as result the node named "ιδέα" ("idea"). As shown in Figure 4 there are more nodes connected to the result. The words associated with the result and the connected nodes, form the object clause. We can reconstruct the object clause of the phrase using information which is included as properties in these nodes.

Specifically, we can use the initial text of the tokens and their position in the sentence. The object clause will include all the words related to these nodes and the tokens between them. In our example the object clause will be "*ιδέες ανεπάρκειας που σχετίζονται με τις συγκεκριμένες κοινωνικές δεξιότητες*" ("*ideas of inadequacy related to specific social skills*"). This phrase is the answer to the question "*Τι έχουν οι πάσχοντες;*" ("*What do the sufferers have*?").

To get answers to the user's questions posed in natural language, we need to process the questions grammatically and express them as Cypher queries. The parse tree of the question is extracted and its components are used to determine the type of the question, the included words and their relationships. These parameters are provided as input to the algorithm for the Cypher query construction. In the sequel we provide two examples:

Question Example 1:

User question: "*Ποιος έχει ιδέες;*" ("*who has ideas*?"). The algorithm determines the subject of the verb being searched and constructs the following query which returns a graph path containing the node of the question and one of the subjects.

match p=(who)-[r1:nsubj]-(VERB{name:'έχω'})-[r2:obj]-(NOUN{name:'ιδέα'}) return p

The query returns the path in Figure 5.

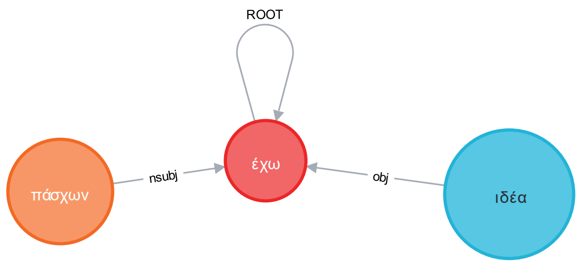

Figure 5: Path returned by the query.

After reconstructing the initial text, the provided answer is "*Οι πάσχοντες έχουν ιδέες*" ("*The sufferers have ideas*").

Question Example 2:

User question: "*Ποιος έχει ιδέες ανεπάρκειας;*" ("*Who makes thoughts that he/she is insufficient?*"). The algorithm creates a more complex query:

match p=(who)-[r1:nsubj]-(VERB{name:'έχω'})-[r2:obj]-(n:NOUN{name:'ιδέα'}) where exists((n)-[:nmod]-(:NOUN{name:'ανεπάρκεια'})) return p

After reconstructing the initial text using query's results, the answer is "*Οι πάσχοντες έχουν ιδέες ανεπάρκειας*" ("*sufferers have ideas of inadequacy*").

Our algorithm includes in the answer that it has constructed up to this point, the nodes which are connected to the node named "ιδέα". So, the final answer becomes, "*Οι πάσχοντες συχνά έχουν ιδέες ανεπάρκειας που σχετίζονται με τις συγκεκριμένες κοινωνικές δεξιότητες*" ("*Sufferers have ideas of inadequacy which are related to specific social skills*").

3.3.1. Complex questions

We provided some basic examples of our system where the initial text given for information retrieval and the related questions were quite simple. Real world documents and real users' questions tend to be more complex. Our QA system should be able to answer complex user's questions. This demands advanced text processing operations to be designed and implemented. These operations include the following:

1. Search for relations between remote nodes indirectly connected.
2. Replace syntax relationships to pronouns with relationships to referred nouns and proper nouns.
3. Add new relationships between nodes that refer to the same entities met in different sentences, paragraphs, or documents.
4. Words and phrases are replaced by synonyms to search and match nodes or paths having the same or similar meaning.
5. Developing of a complementary conversational search algorithm to ask user for additional info in case of partial match.

Operations 1 and 2 have already been implemented and tested while the others are under development. Our system can process successfully complex web text medical documents as the one in Figure 6.

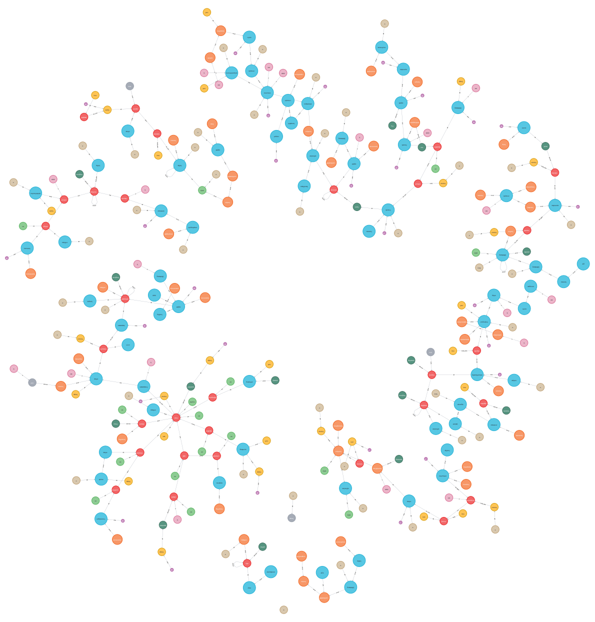

Figure 6: An example graph of a non-trivial document

4. Future Work

The system which has been developed is able to provide precise answers to users' questions. Prerequisite for accurate answers is the construction of the right query based on user's question text. As the entities and their relationships in a question increase, the complexity of the query is increased as well. We will keep on working on query optimization.

We plan to extend our approach on several directions such as the following ones.

- Study and implement the synthesis of answers combining different phrases in a document.
- Perform ranking of possible different candidate right answers.
- Develop methods to deal with ambiguity.
- Perform optimization of the knowledge representation method focusing on semantics and building a knowledge graph.

After the system has been developed, we plan to test it with a big number of documents and users' questions in order to evaluate its accuracy and its efficiency. In future, we will also consider if our approach can be extended with ontologies. We will study if a graph database and its enhanced representation as a knowledge graph can be mapped to an ontology and vice versa.

5. Conclusion

We have designed and developed a medical Question Answering System to be accurate, fast and expandable.

We carry out research on an enriched parse tree navigation utilizing the advantages of a graph database.

The use of graph database provides fast development, it makes it easy to handle complex queries and it can handle big data efficiently.

Our system has been designed with the objective to derive correct and accurate answers to user's queries.

In summary, the advantages our approach are the following. It supports information retrieval from texts for constructing its enhanced parse tree and storing it in a graph database. Our methodology is natural language independent. In its current form, it supports the Greek and the English languages, moreover it can be extended to other natural languages as well. It is also application domain independent. Apart from medicine which is the test domain for our approach it can be used in other domains where natural interaction is required.

Our approach can have diverse applications in healthcare. It can be used in clinical practice for communication and data collection and in clinical decision support. It can be used in hospital management for data management due to the large medical documentation. It can be used for personal health assistants. Finally, it can be used from people for getting health knowledge and in medical education.

6. Acknowledgements

This research work is funded by the Public Investment Program of the Ministry of Education and Religious Affairs of Greece.

7. References

[1] H. A. Pandya and B. S. Bhatt, "Question Answering Survey: Directions, Challenges, Datasets, Evaluation Matrices," *arXiv preprint arXiv:2112.03572*, p. 2, 2021.

[2] Z. Jiang, C. Chi and Y. Zhan, "Research on medical question answering system based on knowledge graph," *IEEE Access*, vol. 9, pp. 21094-21101, 2021.

[3] Q. Shuai, "Research on Intelligent Question Answering System Based on Medical Knowledge Graph," *IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 1, no. IEEE, 2019, 2019.

[4] K. Papantoniou and Y. Tzitzikas, "NLP for the Greek language: a brief survey," *11th Hellenic Conference on Artificial Intelligence*, pp. 101-109, 2020.

[5] E. Marakakis, H. Kondylakis and A. Papakonstantinou, "APANTISIS: A greek question-answering system for knowledge-base exploration," in *Strategic Innovative Marketing: 5th IC-SIM, Athens, Greece 2016*, 2017.

[6] H. Kondylakis, D. Tsirigotakis, G. Fragkiadakis, E. Panteri, A. Papadaki, A. F. and E. Tzagkarakis, "R2D2: A dbpedia chatbot using triple-pattern like queries," *Algorithms*, vol. 13, no. 9, p. 217, 2020.

[7] M. Mountantonakis, B. Michalis, M. Loukas and T. Yannis, "Tiresias: Bilingual Question Answering over DBpedia," 2022.

[8] D. Chen, A. Fisch, J. Weston and A. Bordes, "Reading wikipedia to answer open-domain questions," *arXiv preprint arXiv:1704.00051*, 2017.

[9] Y. Tang, H. Han, X. Yu, J. Zhao, G. Liu and L. Wei, "An Intelligent Question Answering System based on Power Knowledge Graph," *2021 IEEE Power & Energy Society General Meeting (PESGM)*, pp. 1-5, 2021.

[10] X. Huang, J. Zhang, D. Li and P. Li, "Knowledge Graph Embedding Based Question Answering," *Proceedings of the twelfth ACM international conference on web search and data mining*, pp. 105-113, 2019.

[11] E. Zimina, J. Nummenmaa, K. Jarvelin, J. Peltonen, K. Stefanidis and H. Hyyrö, "GQA: Grammatical Question Answering for RDF data," *Semantic Web Challenges: 5th SemWebEval Challenge at ESWC 2018, Heraklion, Greece, June 3–7, 2018, Revised Selected Papers 5*, vol. 927, pp. 57-61, 3-7 June 2018.

[12] E. Partalidou, E. Spyromitros-Xioufis, S. Doropoulos, S. Vologiannidis and K. Diamantaras, "Design and implementation of an open source Greek POS Tagger and Entity Recognizer using spaCy," *IEEE/WIC/ACM International Conference on Web Intelligence*, 2019.

[13] J. Guia, V. G. Soares and J. Bernardino, "Graph Databases: Neo4j Analysis," *ICEIS*, vol. 1, pp. 351-356, 2017.

[14] J. Stothers and A. Nguyen, "Can Neo4j replace PostgreSQL in healthcare?," *AMIA Summits on Translational Science Proceedings*, p. 646, 2020.

[15] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg and P. Selmer, "Cypher: An Evolving Query Language for Property Graphs," *Proceedings of the 2018 international conference on management of data*, pp. 1433-1445, 2018.