ESKİŞEHİR TECHNICAL UNIVERSITY JOURNAL OF SCIENCE AND TECHNOLOGY A- APPLIED SCIENCES AND ENGINEERING

2019, 20(3), pp. 274 - 295, DOI: 10.18038/estubtda.624498

NATURAL LANGUAGE QUESTION ANSWERING SYSTEM OVER LINKED DATA

Abdullah Talha KABAKUŞ [1, *], Aydın ÇETİN [2]

[1] Department of Computer Engineering, Faculty of Engineering, Düzce University, Düzce, Turkey
[2] Department of Computer Engineering, Faculty of Technology, Gazi University, Ankara, Turkey

ABSTRACT

Linked Data project is aimed to give more details on any subject through the big knowledge bases defined on the web. In this context, knowledge bases offer endpoint service user interfaces to query their data. Because of the SPARQL query language limitation of these knowledge bases, a significant number of web users are unable to benefit from these services. In this paper, an English natural language question answering system over Linked Data is proposed in order to eliminate this limitation. The proposed system's main processes can be listed as follows: (1) Extracting Part-Of-Speech (POS) tags, (2) pattern extraction & preparing appropriate SPARQL queries, (3) executing user queries & displaying the results. The features which are not provided by the endpoint services of knowledge bases such as dynamic paging, voice search and answer vocalization which make the usage of the proposed system to be possible by the visually-impaired web users, question-answer caching, social media integration, and live spell checking are proposed. According to experimental results, the proposed system's question answering performance is improved between 2 and 12 times through the type of natural language question thanks to the question-answer caching mechanism.

Keywords: DBpedia, Natural Language Processing, Speech Recognition, Semantic Web

1. INTRODUCTION

Tim Berners-Lee describes Semantic Web aka *Web 3.0* as "The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [1]. Another semantic web definition from PCMag is "a place where machines can read Web pages much as we humans read them, a place where search engines and software agents can better troll the Net and find what we're looking for" [2]. As the definitions emphasize, the semantic web focuses on meaningful, relational data on the web by extending all existing web functionalities. The semantic web is built on Resource Description Framework (RDF) triples to define data a form like *<Turkey, populationCensus, 67803927>*. Simple Protocol and RDF Query Language (SPARQL) is an RDF query language, that is, a query language for databases, able to retrieve and manipulate data stored in RDF format [3], [4].

1.1. Linked Data Project

Linked Data is the project of combining huge structured databases like DBpedia[1], YAGO[2], Freebase[3], FOAF[4], GoPubMed[5] that is started earlier of 2007. Technically, Linked Data refers to data published on the web in such a way that it is machine-readable, its meaning is explicitly defined, it is linked to other external data sets, and can, in turn, be linked to from external datasets [5]. The latest version of Linked Data cloud diagram (Figure 1) which is updated on 30th May 2018 has 1,186 datasets and 188 million

[1] https://dbpedia.org
[2] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/
[3] https://developers.google.com/freebase/
[4] http://www.foaf-project.org
[5] https://gopubmed.org/web/gopubmed/

*Corresponding Author: talhakabakus@duzce.edu.tr
Received: 19.12.2018 Published: 26.09.2019

triples so far [6], [7]. Therefore, Linked Data is more structured and machine processable; applications can traverse this web of data, easily find useful data and pinpoint the right information [5].

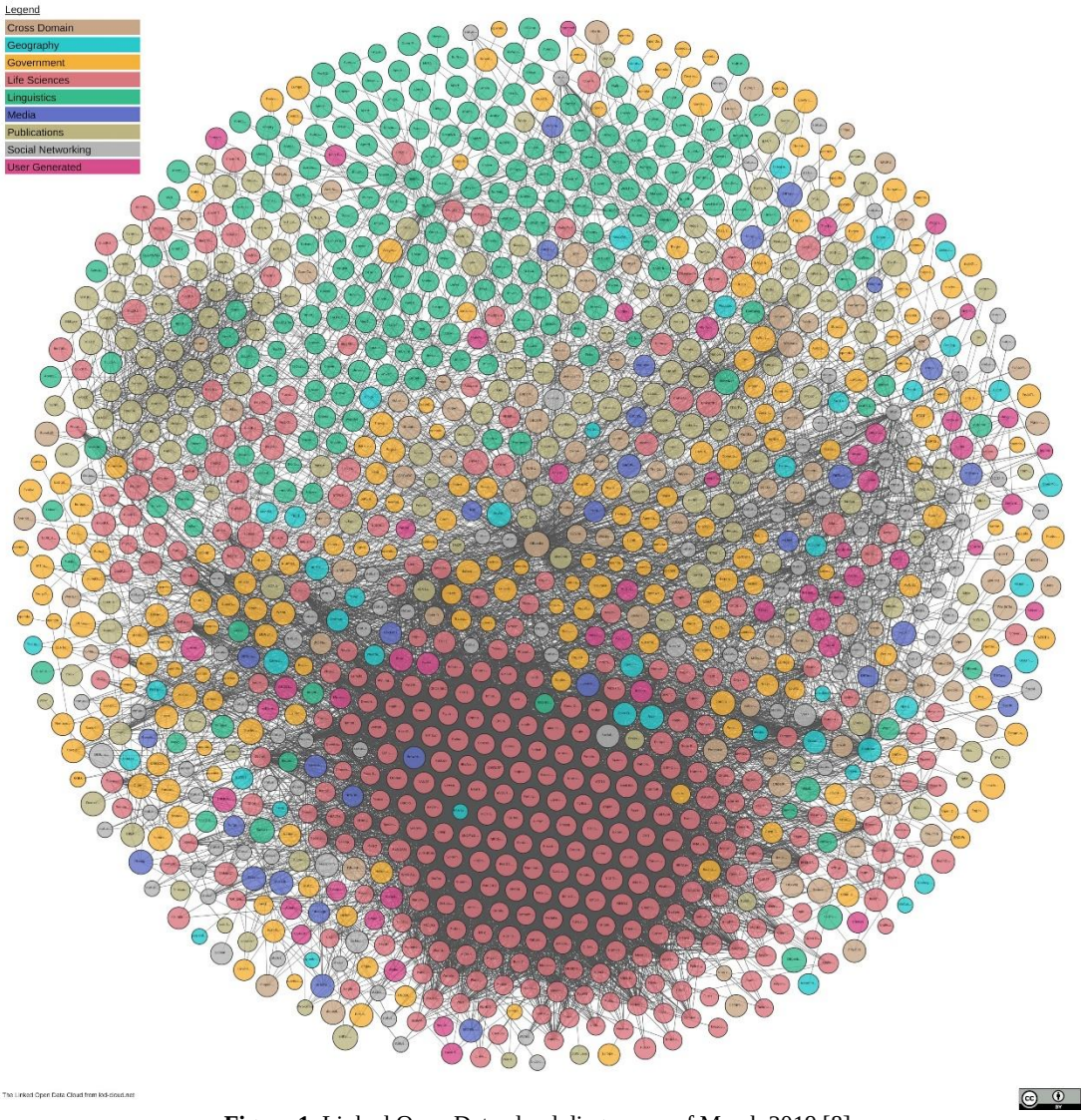

Figure 1. Linked Open Data cloud diagram as of March 2019 [8]

1.2. DBpedia

DBpedia data is created by converting infobox information of Wikipedia[6] into well-defined and structured form. DBpedia allows you to ask sophisticated queries against Wikipedia and to link other datasets on the Web to Wikipedia data [8]. DBpedia project is started in 2007 and developed by a collaboration of the Free University of Berlin, University of Leipzig and OpenLink Software [9]. DBpedia is one of the centrally linked data datasets in the Linked Open Data project [10]. According to DBpedia Wiki, DBpedia knowledge base (English version) contains more than 4.58 million things, including 1,445,000 persons, 735,000 places, 123,000 music albums, 87,000 films and 19,000 video games, 241,000 organizations, 251,000 species and 6,000 diseases [11]. In addition to this, DBpedia provides knowledge in 125 languages. All versions together describe 38.3 million things. The whole

[6] https://wikipedia.org

dataset consists of 3 billion pieces of information which are called as *RDF triples*. DBpedia provides an endpoint to query triples using SPARQL [12].

The main components of the DBpedia knowledge extraction framework can be listed as follows: *PageCollections* which are an abstraction of local or remote sources of Wikipedia articles, *Destinations* that store or serialize extracted RDF triples, *Extractors* which turn a specific type of wiki markup into triples, *Parsers* which support the extractors by determining data types, converting values between different units and splitting markup into lists. *ExtractionJobs* group a page collection, extractors and a destination into a workflow. The core of the framework is the *Extraction Manager* which manages the process of passing Wikipedia articles to the extractors and delivers their output to the destination. The *Extraction Manager* also handles Unified Resource Identifier (URI) management and resolves redirects between articles. Figure 2 illustrates an overview of DBpedia components.

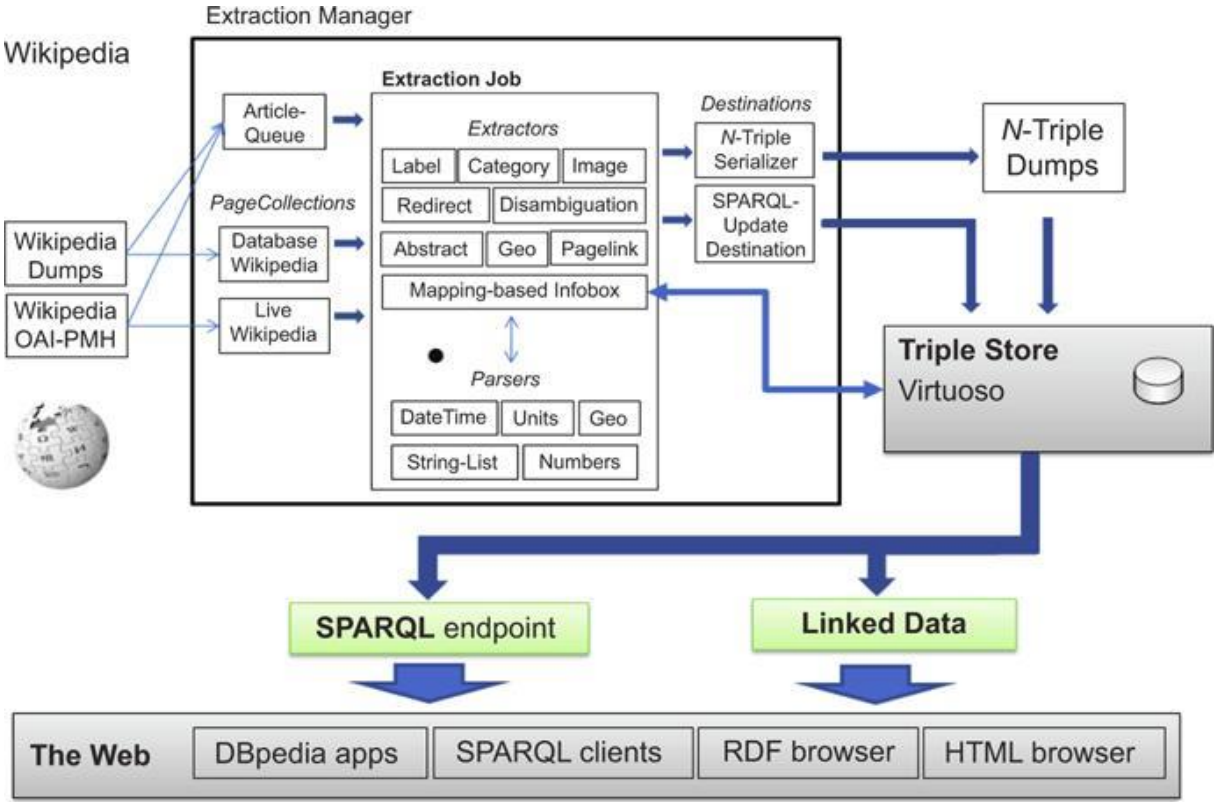

Figure 2. An overview of DBpedia components [11]

The web is based on human-computer collaboration to provide interactive media. Several reports show that one of the major reasons for the usage of the web by its users is finding answers to their questions [13]–[15]. The proposed system uses semantic web technologies in order to find answers to users' questions. Most semantic web knowledge bases provide endpoint services in order to open their data. But these endpoint services accept SPARQL as query input language which limits its user crowd. To overcome this limitation, we propose a semantic web-based question answering system that offers to query data over these endpoints through natural language questions.

2. RELATED WORK

Ostuni et al. [16] present a mobile movie recommendation system based on Linked Data named Cinemappy which is developed for Android-based mobile devices. All information related to movies such as genres, actors, directors, etc. are extracted from DBpedia. They use both English and Italian

versions of DBpedia ontology while making recommendations. This approach may increase the complexity of the system. Because of these two ontologies come from the same origin, we believe benefiting from one of these ontologies and presenting information in different languages will make the system faster and more consistent. Also, with this approach, all limitations based on languages will be eliminated too.

Ell et al. [17] present a system named SPARTIQULATION which mainly works to translate SPARQL queries into natural language expressions which can be readable and easily understandable by end users. SPARTIQULATION has a different approach than other systems. This system accepts everything as "thing" and types of answers are determined by limitations on SPARQL queries. One of the major disadvantages of this system is that it just uses RDF and RDF Schema (RDFS) which are limited compared to Web Ontology Language (OWL) and Friend Of A Friend (FOAF). We benefit from all resource description languages including RDF, RDFS, OWL, FOAF and even DBpedia's own ontology. SPARTIQULATION translates semantic web resources into natural language expressions to vocalize results. This approach can be inefficient especially for complex resources. Instead of this approach, we get all the information about resources through DBpedia which is regularly checked and updated by people who is working for it. Due to our system uses answers directly from DBpedia instead of transforming raw data into natural languages expressions, we believe that our approach is more stable and consistent than theirs.

Lopez et al. [18] present a question answering system named PowerAqua. The main difference of PowerAqua is it combines multiple data sources *(knowledge bases)* while answering questions. This approach has a heterogeneity threat because of different ontologies are used by knowledge bases. It is not always possible to associate one property with another one defined on other knowledge bases. So this can cause inconsistent and low-quality results. The proposed system uses only DBpedia as information source while answering questions. This approach guarantees the homogeneity of data and prevents the system from low-quality results. QAKiS [19] utilizes a relation-based match that tries to match fragments of questions and relational textual patterns which are automatically collected from Wikipedia.

Damljanovic et al. [20] present system named FREyA. The main disadvantage of this system is its need for supervision to train it to make it ready to start answering questions. System's question answering mechanism capability relies on users' feedbacks. This approach can cause inconsistent results through the wrong supervision or cyber attacks in the form of user. Also, these types of systems always behave differently in respect of users who trained it. Our system isn't in need of training as it only benefits from users' feedback for their advantages while making recommendations. Our system always behaves the same and its question answering mechanism is not affected by users. So our system can be accepted as more secure and consistent.

Ferrucci et al. [21] present a well-known and complex system named Watson owned by IBM[7]. The major difference of Watson from other systems is that the system decomposes questions into many sub-cues and finally merges and ranks the results. According to IBM, Watson has more than 100 different techniques are used to analyze natural language, identify sources, find and generate hypotheses, find and score evidence, and merge and rank hypotheses [22].

Unger & Bühmann [23] present a system based on template queries. This approach has a limitation over different types of questions. Templates can be useful to answer questions which exactly fit predefined templates. However, predefined patterns limit the variety of target questions since it is not always possible to find the related property (predicate) to the extracted verb of sentence on knowledge bases. Our system gives some related information about found answers even it is not able to find equal property

[7] https://ibm.com

for the extracted verb. We have used both predefined patterns and instantly created queries through the similarity between DBpedia ontology and extracted natural language questions while preparing SPARQL queries. Our approach aims to answer as many questions as possible.

Pythia [24] has a different approach than most of question answering systems; it is based on linguistic analysis. The system constructs its own lexicon and processes complex questions linguistically in this way. Our system is based on DBpedia's ontology and tries to find the answer through its ontology. Similar to Pythia, ORAKEL [25] is also an ontology-based question answering systems where ontologies play a central role during the interpretation of questions. The most major advantage of these systems is that since the corresponding constructs are covered by the grammar, they are able to deal with the complex questions. The clearest drawback of these approaches is that they fail when a question cannot be parsed by the grammar [26].

Yahya et al. [27] present a method that segments questions into phrases, then maps these phrases into semantic entities, classes, and relations. They check if two semantic items are compatible or not through the signatures defined in each relation. They group semantic items into triples, then they construct a SPARQL query. Their system tries to execute complex queries in this way.

Kabakus & Dogdu [10] present a static approach which is built on predefined patterns while processing natural language questions which can fail to answer different types of questions due to their static (predefined) natural language processing mechanism. We benefit from both predefined patterns and DBpedia ontology while creating appropriate SPARQL queries. Also, they don't provide a caching mechanism while answering natural language questions. Even a question is just asked milliseconds ago, they process all steps again. We store questions with their answers for one month to prevent outdated answers. During this duration, answers of stored questions are loaded from the database instead of processing whole steps. This approach makes it possible to answer recently asked questions a lot faster than before. Also, their system doesn't offer dynamic result count which lets the user choose how many answers they want to be listed in each page. This feature lets users specify a range of answers which change through questions. The conducted experiments have demonstrated that this feature becomes very useful for pinpoint questions who have fewer answers; in this case, the proposed system answers questions faster than before. We provide an optional user registration in order to keep a history of each user and make search recommendations through their previous searches. Otherwise, the history of searches would be limited to users' sessions.

Graph-based approaches such as top-k exploration [28] and the approach proposed by Shekarpour et al. [29] interpret a natural language question by mapping its elements to the entities of knowledge-base. The major drawback of these approaches is that exhaustive searches on graphs make the system unfeasible. Also, complex questions which need aggregation cannot be answered when they do not have a one-to-one correspondence to a path in the graph [26].

Unger et al. [26] propose a survey about the approaches focuses on question answering over Linked Data. They summarize the main approaches, available tools and resources, benchmarks, and evaluation campaigns. The motivation point of this study is that none of these related systems use speech recognition to improve search speed in more user-friendly approach and make it available to be used by visually-impaired people as our system does. With search by voice and answer vocalization features, we offer a complete natural language question answering system for visually-impaired people. Another drawback of these systems is they don't offer immediate spell checking to prevent typing errors sourced from end users. Question and answer caching mechanism is another key point escaped notice by the related works. Because of the answer's target variety changes through the question, we let users set answer count (per page) to be retrieved by the system. To the best of our knowledge, our system is the first one that offers this flexibility. Considering rich information defined in social media, the proposed

system is integrated with the most popular and widely used social media platforms to give more detailed and up-to-date information about found answers.

2. METHODOLOGY

The proposed system has three main processes as follows as it is presented in Figure 3:

(1) Extracting Part-Of-Speech (POS) tags from natural language questions
(2) Pattern Extraction & preparing appropriate SPARQL queries
(3) Execute user queries & display results in the web user interface

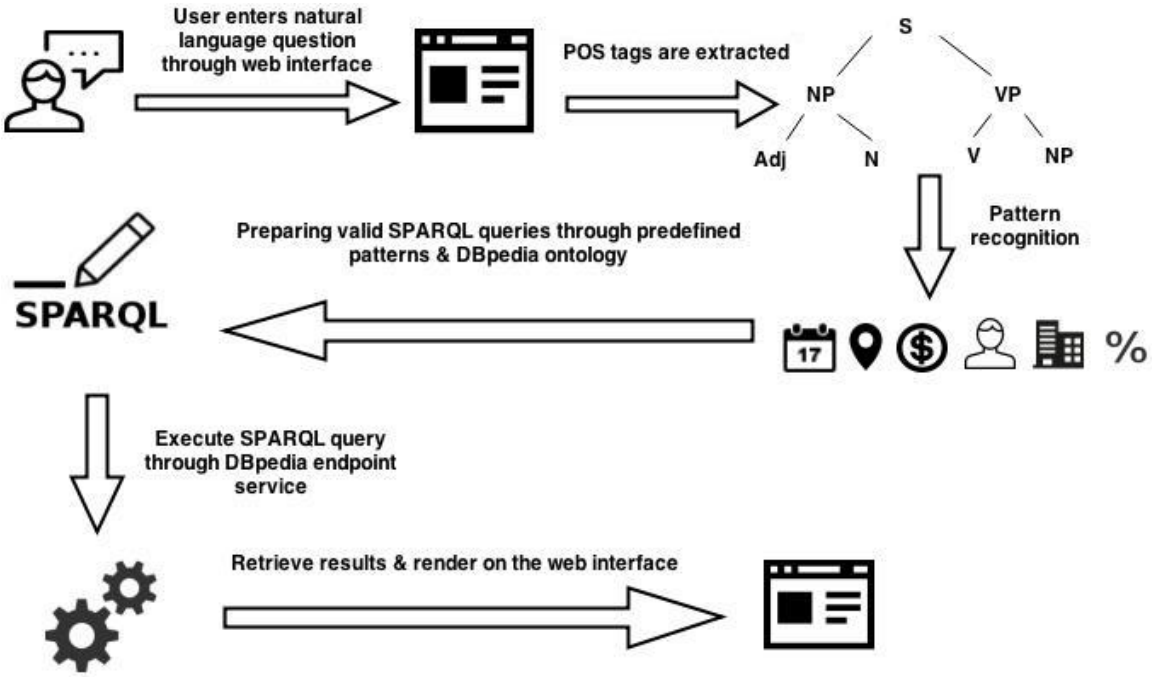

Figure 3. Illustration of the proposed system's methodology

3.1. Extracting POS Tags

The common language of all users of the web is natural language. Therefore, the first task of the proposed system is letting users ask questions using natural language. Open-source Apache OpenNLP[8] library is used to extract natural language questions' POS tags. By utilizing this library, natural language texts are parsed and components of sentences are extracted. Apache OpenNLP is able to extract sentences into the parts through the predefined POS tagset [30].

3.2. Pattern Extraction & Preparing Appropriate SPARQL Queries

After POS tags are extracted, the pattern matching process starts. The proposed system offers a scalable pattern definition, so a number of patterns can be increased by time easily. Defined patterns and their excepted answer types are listed in Table 1.

[8] https://opennlp.apache.org

Table 1. Defined patterns & their expected answer types

Question Pattern	Expected Answer Type
Who/whom	Person
When	Date
Where	Place
What	Everything
Find	Place
Locate	A location that contains a latitude and longitude couple

Main reasons behind using Not Only SQL (NoSQL) databases can be listed as its better performance [31] against big data and better scalability. A great example of a NoSQL database is Facebook's implementation (Cassandra) which is capable of handling more than 100 million users continuously [32]. Also, NoSQL databases store data in JSON (JavaScript Object Notation) format which is the same data format that JavaScript uses. Storing data in JSON format makes it possible to use stored data at client-side (web user interface) without any conversion process. Therefore, the highly popular NoSQL database system MongoDB[9] is used to store irregular verbs to determine if the extracted verb is regular or irregular. Each extracted verb is searched in the database. If the extracted verb is found in the database that means the verb is irregular. After verb is found, associated DBpedia property predicates are searched in whole DBpedia ontology through DBpedia ontology file using Apache Jena[10]. Levenshtein Distance is a string metric for measuring the difference between two sequences. A string comparison based on the Levenshtein Distance algorithm is applied to property predicates to find the most similar one to the found verb.

Each RDF triple contains a subject, a predicate, and an object. Before creating appropriate SPARQL query, we need to find the object of triple properly. After we determined the type of answer, the object of the sentence must match with this defined type. Apache OpenNLP provides a set of training data to obtain the type of objects. This is a very critical process in order to prevent object mismatches in complex sentences. With this approach, we check the detected object twice. Training data and their expected types are listed in Table 2.

Table 2. Training data and their expected types

Training Data	Expected Type
en-ner-date.bin	Date
en-ner-location.bin	Location (city, country)
en-ner-money.bin	Money
en-ner-organization.bin	Organization/company
en-ner-percentage	Percentage
en-ner-person	Person
en-ner-time.bin	Time

Once the predicate (property) and object are found (as the examples of properties with their calculated distance values to the verb of natural language question "*produce*" are listed in Table 3), the associative SPARQL query is constructed. A SPARQL query is constructed through both predefined patterns and extracted POS tags. Both query results are merged (using *UNION* keyword) and a final query is sent to the DBpedia endpoint service.

[9] https://mongodb.org
[10] https://jena.apache.org

Table 3. The DBpedia properties and their distance values calculated using the Levenshtein distance algorithm

DBpedia Property	Levenshtein Distance of the DBpedia Property from the Detected Verb "*produce*"
producer	1
coProducer	4
coExecutiveProducer	13
wineProduced	6
executiveProducer	11

3.3. Execute Queries & Display Results in the Web User Interface

Once the appropriate SPARQL query is constructed, the system tries to find the answer(s) of asked question through the DBpedia endpoint service. Apache Jena is an open source and Java-based semantic web framework provides an advanced API to use different data sources including files, databases, and URLs or a combination of these [10]. As all questions with their answers are stored in the database for one month, before executing SPARQL queries through DBpedia endpoint service, we look up them if they exist in our database. If the question already exists in our database, the answer(s) of question which is a Wikipedia reference ID is retrieved from the database and up-to-date details are get through the DBpedia endpoint service. Otherwise, we execute the constructed SPARQL query through the DBpedia endpoint service and store it with its answer in the database. Figure 4 visually explains the question and answer the caching mechanism of the proposed system.

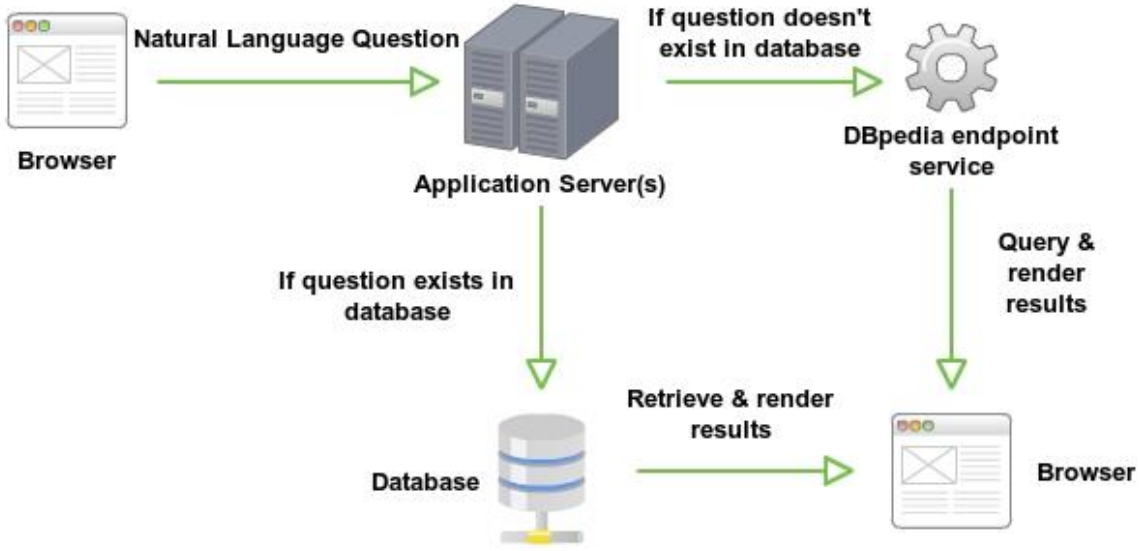

Figure 4. A block diagram of the proposed system's question and answer caching mechanism

PoS/NER is used to match detected PoS tags with the parts of RDF triples. For the example natural language question "*How tall is Michael Jordan?*", the generated PoS tags for the natural language question are as follows: `(SBAR (WHADVP (WRB How)) (S (NP (NN tall)) (VP (VBZ is) (NP (NNP Michael) (NNP Jordan)))))`. For this natural language question, the returned JSON data which is returned from the SPARQL endpoint is listed in Table 4:

Table 4. The JSON data which is returned from the SPARQL endpoint for the natural language question "*How tall is Michael Jordan*?"

Key	Value
uri	http://dbpedia.org/resource/Michael_Jordan
abstract	Michael Jeffrey Jordan (born February 17, 1963), also known by his initials, MJ, is an American retired professional basketball player. He is also a businessman, and principal owner and chairman of the Charlotte Hornets. Jordan played 15 seasons in the National Basketball Association (NBA) for the Chicago Bulls and Washington Wizards. His biography on the NBA website states: "By acclamation, Michael Jordan is the greatest basketball player of all time." Jordan was one of the most effectively marketed athletes of his generation and was considered instrumental in popularizing the NBA around the world in the 1980s and 1990s. Jordan played three seasons for coach Dean Smith at the University of North Carolina. As a freshman, he was a member of the Tar Heels' national championship team in 1982. Jordan joined the NBA's Chicago Bulls in 1984 as the third overall draft pick. He quickly emerged as a league star, entertaining crowds with his prolific scoring. His leaping ability, demonstrated by performing slam dunks from the free throw line in slam dunk contests, earned him the nicknames "Air Jordan" and "His Airness". He also gained a reputation for being one of the best defensive players in basketball. In 1991, he won his first NBA championship with the Bulls, and followed that achievement with titles in 1992 and 1993, securing a "three-peat". Although Jordan abruptly retired from basketball before the beginning of the 1993–94 NBA season to pursue a career in baseball, he returned to the Bulls in March 1995 and led them to three additional championships in 1996, 1997, and 1998, as well as a then-record 72 regular-season wins in the 1995–96 NBA season. Jordan retired for a second time in January 1999, but returned for two more NBA seasons from 2001 to 2003 as a member of the Wizards. Jordan's individual accolades and accomplishments include five Most Valuable Player (MVP) Awards, ten All-NBA First Team designations, nine All-Defensive First Team honors, fourteen NBA All-Star Game appearances, three All-Star Game MVP Awards, ten scoring titles, three steals titles, six NBA Finals MVP Awards, and the 1988 NBA Defensive Player of the Year Award. Among his numerous accomplishments, Jordan holds the NBA records for highest career regular season scoring average (30.12 points per game) and highest career playoff scoring average (33.45 points per game). In 1999, he was named the greatest North American athlete of the 20th century by ESPN, and was second to Babe Ruth on the Associated Press's list of athletes of the century. Jordan is a two-time inductee into the Basketball Hall of Fame, having been enshrined in 2009 for his individual career, and again in 2010 as part of the group induction of the 1992 United States men's Olympic basketball team ("The Dream Team"). He became a member of the FIBA Hall of Fame in 2015. Jordan is also known for his product endorsements. He fueled the success of Nike's Air Jordan sneakers, which were introduced in 1985 and remain popular today. Jordan also starred in the 1996 feature film Space Jam as himself. In 2006, he became part-owner and head of basketball operations for the then-Charlotte Bobcats, buying a controlling interest in 2010. In 2015, as a result of the increase in value of NBA franchises, Jordan became the first billionaire NBA player in history and the world's second-richest African-American.

Apache Jena library is used to execute these SPARQL queries and retrieve results as Java objects (RDFNode). Retrieved results are converted into JSON objects to display in the web user interface. For web user interface development, highly popular and open source Rich Internet Application (RIA) development library named Sencha Ext JS 4[11] is used because of its support for Object-Oriented Programming approaches (OOP) and Model-View-Controller (MVC) architecture at client-side. Model–View–Controller architecture is a design approach of dividing the software into Model, View and Controller component to better control the software quality with respect to processing, and interface design [33]. Major benefits of MVC architecture can be listed as enhanced maintainability and extensibility of the system, potential multiple views of the same model, pluggable views, and controllers, synchronized views, and re-usability [34]. Client-side and server-side implementations are completely separated in order to provide better performance with minimizing server calls. All the user interactions, validations, controls are fulfilled at client-side without distracting the server. For all of these benefits, the client-side of the system is implemented using Sencha Ext JS in order to create rich, powerful and modern user interface widgets with cross-platform browser compatibility. The user interface of the proposed system that shows a sample search with its results is presented in Figure 5.

[11] https://www.sencha.com/products/extjs/

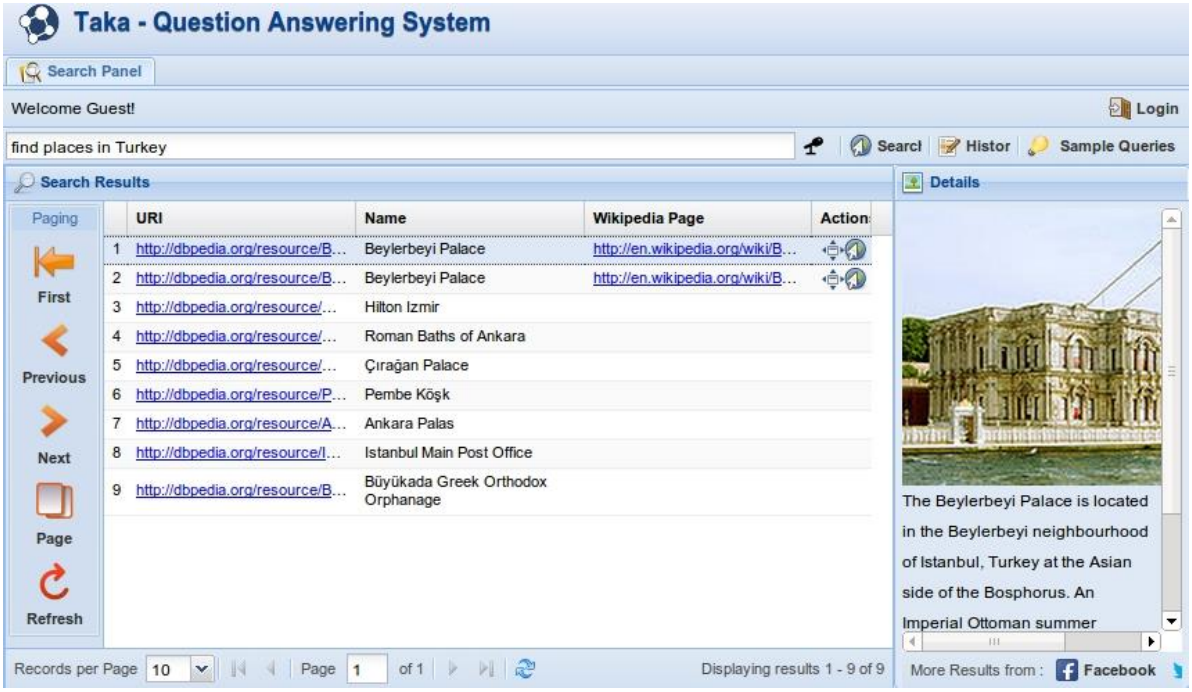

Figure 5. The web user interface that shows a sample search and its results

The result list user interface changes dynamically through the result types. If a result contains a latitude and longitude couple, then a map icon appears in the row. When it is clicked, the location is shown on the map using an open source JavaScript mapping library named LeafletJS[12] with gathering data from Cloudmade[13] as it is shown in Figure 6.

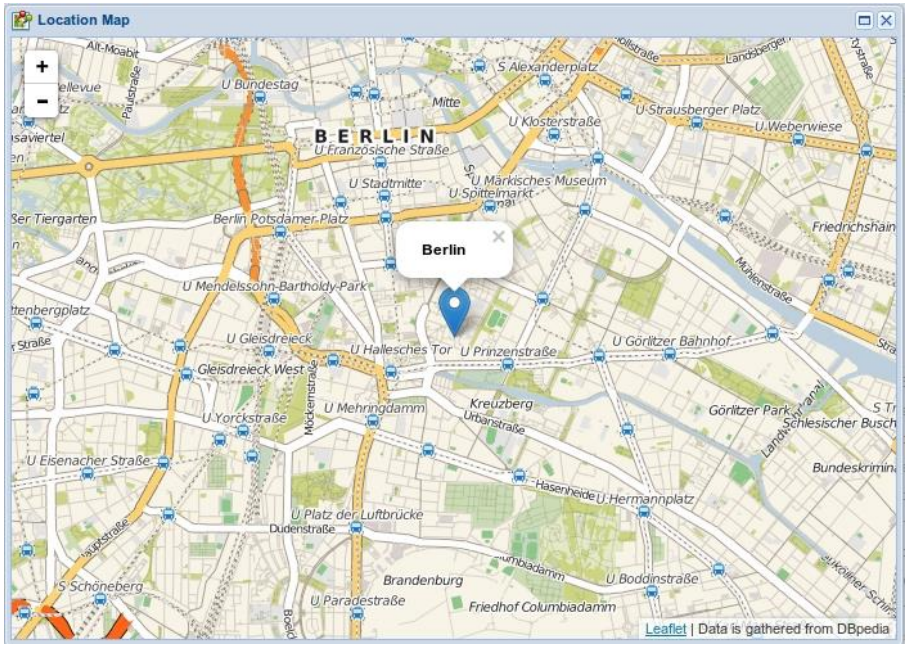

Figure 6. The window that shows location on the map through the data defined in DBpedia

[12] https://leafletjs.com
[13] https://cloudmade.com

Similarly, if the found DBpedia resource (*the answer of the question*) has homepage website URL and Freebase definition that is available on DBpedia, the proposed system is able to show further information defined on these external links as it is shown in Figure 7.

Figure 7. The user interface to show further information defined on external links

If the user cannot find any results through the search query, a recommended question list retrieved from the database is shown to guide the user to see similar answered questions as it is shown in Figure 8. The recommendations are made through (1) the detected target types of questions listed in Table 1 which is called as '*semantic similarity*', and (2) the calculated syntactic similarity between the asked question and stored ones. The semantic similarity process lets the system prioritize similar questions in term of its target types while looking for a recommendation. For the similarity measurement, the Levenshtein Distance algorithm is used and the results are sorted through those similarity values.

Figure 8. The search recommendations window

3.3.1. Speech Recognition and Answer Vocalization

The web is a common area of people and the number of visually-impaired or hearing-impaired users are growing day by day. In order to cover as many people as possible, speech recognition and answer vocalization are integrated into the proposed system. Speech recognition also lets users ask natural language questions more easily and quickly. These features make the system usable by visually-impaired people.

Speech recognition is applied via a recognition confidence threshold which is a value between 0-1 that determines the acceptable confidence limit. With considering pronunciation flexibilities and hardware deficiencies, it is set to 0.75 which gives out best results at model training. There are some speech

recognition libraries available like CMU Sphinx[14], TalkingJava[15]. The reason behind using Web Speech API[16] is that it completely works on client-side, so it doesn't engage application server with speech recognition requests.

The text-to-speech mechanism is implemented to inform especially visually-impaired people about found answers. Google Translate[17] service is used for text-to-speech conversation. Answers are converted to speech then it is vocalized to users through this service. Proposed system's speech recognition and question-answer vocalization mechanisms are illustrated in Figure 9.

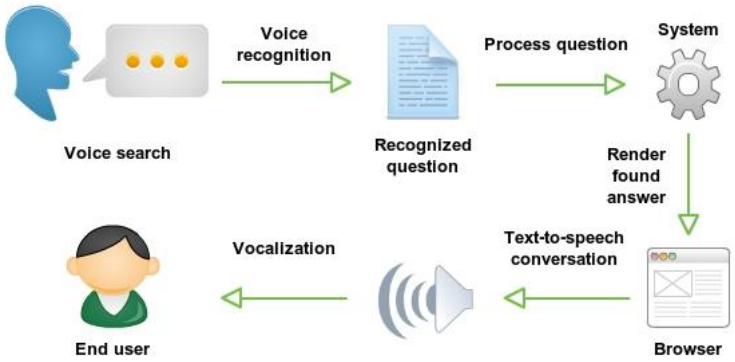

Figure 9. Speech recognition and question answer vocalization mechanism

3.3.2. Spell Check

Users are able to ask their questions with both entering texts and using their voices. Speech recognition is a safer approach to prevent spelling errors because of its predefined database. To prevent typing errors, spell checking mechanism works on client-side that checks as soon as the user enters questions. *Typo.js* which is a JavaScript spell checker that uses Hunspell-style dictionaries is used for spell checking. If there is no spelling error, then the search process continues. Otherwise, a window like shown in Figure 10 is shown to let users correct his/her typing errors with the suggestions that the system offers. This window offers typical operations that most of the text editors provide such as *Ignore*, *Ignore All*, *Replace*, *Replace All*, *Save* and *Cancel*. This approach will not only help the user but also prevents the server from processing wrong spelled questions which cannot be answered by the system.

Figure 10. Spell checking window

[14] https://cmusphinx.github.io
[15] https://www.javaworld.com/article/2075495/java-se/talking-java-.html
[16] https://w3c.github.io/speech-api/speechapi.html
[17] https://translate.google.com

3.3.3. Answer Caching Mechanism

Answers to natural language questions are stored in the database to be used when the same question is asked. This approach eliminates whole natural language processing and preparing appropriate SPARQL query steps for recently asked questions. To keep the results' details up to date, only the reference numbers of answers are stored in the database. So each search returns the latest, updated information about the answer.

Another key point of this caching process is that answers to questions can also be changed by time. In order to prevent the system from retrieving outdated answers from system, the default caching duration is set as one month. When the caching duration time is out, the system automatically purges these outdated answers. Once these purged questions are asked again, the system regards them "*new*" and processes all the steps described before. Table 6 shows performance improvements gained by caching mechanism for answering some sample natural language questions in 3,872 stored questions. As it is listed in Table 5, the proposed system's question answering performance is accelerated between 2-12.5 times regarding target size of answers thanks to the proposed caching mechanism.

Table 5. Some sample results that show performance improvements gained by caching mechanism

Natural Language Question	Response Time (sec.)	Cache Response Time (sec.)	Performance Gain (times)	SPARQL Query
who is Barack Obama?	27	2	12.5	SELECT DISTINCT * WHERE { { ?x rdfs:label "Barack Obama"@en . ?x dbont:abstract ?abstract . ?x rdfs:label ?name . OPTIONAL { ?x dbont:thumbnail ?thumbnail . ?x foaf:isPrimaryTopicOf ?wiki . ?x dbont:wikiPageID ?wikiPageID . ?x foaf:homepage ?homepage . ?x owl:sameAs ?sameAs . FILTER regex(str(?sameAs), '^http://rdf.freebase.com', 'i') } FILTER (lang(?name) = 'en' && lang(?abstract) = 'en') } UNION { } } LIMIT 1 OFFSET 0
find places in Germany	28	3	8.3	SELECT DISTINCT * WHERE { { ?x dbpedia2:locationCountry :Germany . ?x dbpedia2:name ?name . ?x dbont:abstract ?abstract . OPTIONAL { ?x dbont:thumbnail ?thumbnail . ?x foaf:isPrimaryTopicOf ?wiki . ?x dbont:wikiPageID ?wikiPageID . } FILTER(lang(?abstract) = 'en') } UNION { ?y dbont:dfE ?x . ?y rdfs:label ?target . ?x rdfs:label ?name . ?x dbont:abstract ?abstract . OPTIONAL { ?x foaf:isPrimaryTopicOf ?wiki . ?x dbont:wikiPageID ?wikiPageID . ?x dbont:thumbnail ?thumbnail . } FILTER(regex(?target, '^Germany', 'i') && lang(?name) = 'en' && lang(?target) = 'en' && lang(?abstract) = 'en') } } LIMIT 1 OFFSET 0
who produce Saturnight?	22	2	10	SELECT DISTINCT * WHERE { { ?x rdfs:label "Saturnight"@en . ?x dbont:abstract ?abstract . ?x rdfs:label ?name . OPTIONAL { ?x dbont:thumbnail ?thumbnail . ?x foaf:isPrimaryTopicOf ?wiki . ?x dbont:wikiPageID ?wikiPageID . ?x foaf:homepage ?homepage . ?x owl:sameAs ?sameAs . FILTER regex(str(?sameAs),

				'^http://rdf.freebase.com', 'i') } FILTER (lang(?name) = 'en' && lang(?abstract) = 'en') } UNION { ?y dbont:producer ?x . ?y rdfs:label ?target . ?x rdfs:label ?name . ?x dbont:abstract ?abstract . OPTIONAL { ?x foaf:isPrimaryTopicOf ?wiki . ?x dbont:wikiPageID ?wikiPageID . ?x dbont:thumbnail ?thumbnail . } FILTER(regex(?target, '^Saturnight', 'i') && lang(?name) = 'en' && lang(?target) = 'en' && lang(?abstract) = 'en') } } LIMIT 1 OFFSET 0
how tall is Michael Jordan?	29	4	6.25	SELECT DISTINCT * WHERE { { ?x rdfs:label "Michael Jordan"@en . ?x foaf:isPrimaryTopicOf ?wiki . ?x dbo:wikiPageID ?wikiPageID . ?x dbo:abstract ?abstract . OPTIONAL { ?x dbo:thumbnail ?thumbnail . ?x dbpedia2:name ?name . } FILTER (lang(?abstract) = 'en') } UNION { } } LIMIT 1 OFFSET 0
where is Paris?	20	6	2.33	SELECT DISTINCT * WHERE { { ?c rdf:type dbont:Country . ?x dbont:country ?c . ?x rdfs:label ?name . ?x dbont:abstract ?abstract . ?x dbont:thumbnail ?thumbnail . ?name <bif:contains> "Paris" . ?x foaf:isPrimaryTopicOf ?wiki . ?x dbont:wikiPageID ?wikiPageID . FILTER(lang(?abstract) = 'en' && lang(?name) = 'en') } UNION { } } LIMIT 1 OFFSET 0

Since we store all questions with their answers on the database, we have evaluated the proposed system's search performance through different sizes of stored questions to reveal its performance change. For this purpose, a service is implemented to create and store new questions. The proposed system's search performance is tested against 32, 3,872, 42,592, 468,512 and 5,153,632 records in order to reveal its performance in term of elapsed time to answer a question.

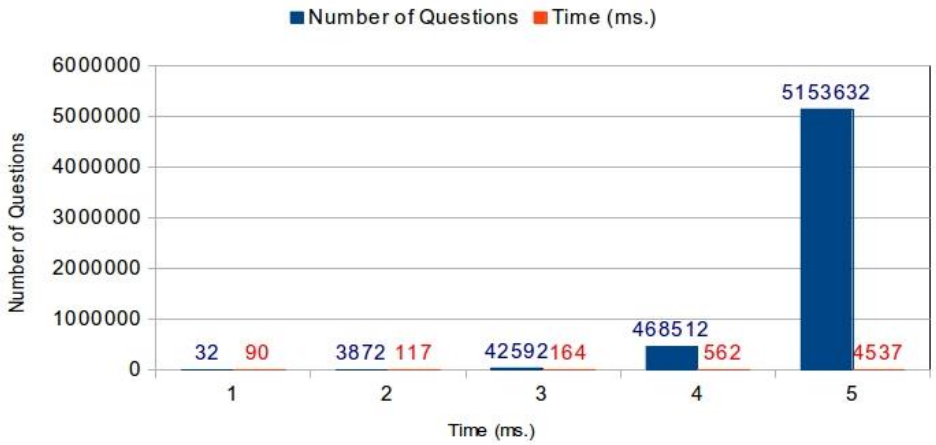

Figure 11. The proposed system's search performance against the different size of data

As the experimental results are shown in Figure 11, the proposed system is able to find a question inside more than 5.1 million questions in 4.5 seconds. This high performance's main reasons can be listed as:

1. The type of database: NoSQL databases offer high performance against big data because of its lightweight structure
2. We only store the DBpedia reference values of answers – so they are just integer values and take minimal space on the database

3.3.4. Social Media Integration

According to the several researches by companies Go-Globe and Social Jumpstart in every single minute, 2 million YouTube[18] videos are viewed, 700 thousand Facebook[19] statuses are sent, 700 thousand searches are queried through Google[20], 175 thousand tweets are sent through Twitter[21] [35], [36]. According to Twitter officials, Twitter has well over 200 million active users creating over 400 million Tweets each day [37]. Today social media's power on people and also organizations cannot be underestimated. Social media is the place of sharing and getting knowledge with all over the world. As the statistics above show the huge amount of information is shared on social media every day. Respect to this power of social media, we integrated major social media platforms Facebook, Twitter and Google to show more details about answers to users and made it available with one click as it can be seen in Figure 12.

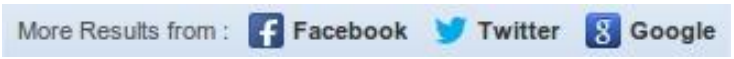

Figure 12. The social media search toolbar

4. RESULTS AND DISCUSSION

Natural language questions have endless types and each question can be asked in many forms despite they have all the same answer. So it seems impossible to develop a system that senses all of these varieties and answers all of them. To demonstrate this, let's take the "*where is the largest city in USA?*" natural language question. We can ask the same question like "*where is the biggest city in USA?*". Because of both of these questions have the same answers, the system must be able to identify this equality and construct the same appropriate SPARQL query for these questions. Appropriate SPARQL query for these questions should be: |*Subject: USA*| |*Predicate: dbprop:largestCity*| |*Object: ?x*|. To get over all of these varieties, we use both predefined patterns that recognize well-known question types and instantly constructed triples through extracted POS tags. Even if our system cannot answer a question combining these approaches, related information about the answer is presented to users. In this way, we don't let any question to be unanswered and try to answer as many questions as possible. Semantic web technologies offer to associate different semantic resources using OWL. OWL offers a property (*owl:sameAs*) to define equivalent resources on the semantic web which is commonly used by knowledge bases in Linked Data to associate their resources with other knowledge bases.

A user interface similar to Wikipedia's infobox is developed to give more information about found answers is shown to users. (Figure 13) A thumbnail, abstract, home page of the answer (if it exists) and social media toolbar are displayed in this interface. Thumbnails are defined as URLs on DBpedia with *dbpedia-owl:thumbnail* property and rendered in this interface. When the user selects an answer on the list, answer details are shown in this panel and are vocalized. Answer list user interface changes dynamically through the type of answer. Additionally, the home page link of the answer, an interactive map that shows location and Freebase definition window becomes visible when it exists. Social media toolbar is also embedded for each answer to give more up-to-date and detailed information about answers on social media including Facebook, Twitter, and Google.

[18] https://youtube.com
[19] https://facebook.com
[20] https://google.com
[21] https://twitter.com

Figure 13. The user interface that shows details of each answer

Voice search and answer vocalization approaches offer visually-impaired people to use this system and learn answers to their natural language questions. A keyboard shortcut is assigned to start the voice search. Because of voice search is the simplest, quickest way to ask natural language questions, this approach eases question asking not for only visually-impaired users; but also other users too. Voice search is also safer than typing due to recognition is limited to a predefined list. The found answers are vocalized to users when they select each answer. Keyboard navigation is enabled in order to make visiting between answers available to visually-impaired people. Because different implementations are used for web browsers, the proposed system's voice search mechanism is only available for users of Google Chrome 25+ which is currently the only platform supports Web Speech API. Despite Google Chrome's huge popularity among users of the web[22], this is a limitation for users of other browsers.

The proposed system provides performance raising features to the DBpedia endpoint service through pagination, dynamic paging and caching supports as it is described before. Despite that we cache answers to questions, answer details are always up to date because we only store references (*unique numeric values defined by DBpedia*) of answers. The system also offers live type checking in order to prevent the system from wrong-spelled questions that system will not be able to answer. Therefore, these unnecessary question answer requests will be eliminated and the system will not try to answer them. This approach also helps users to correct their spelling errors and guide them to right. Integrated recommendation mechanism makes recommendations to users based on previous searches when the system fails to find answers to the asked question. With this approach, users are guided to ask the right questions to find answers they are looking for. Another performance improvement that our system offers is a dynamic result (*answer*) count (*per page*) option. With this feature, users are able to set how many answers they want to see on each page. This lets users decrease result count to get answers quickly for pinpoint questions. Also, they can easily increase page size via web user interface if their questions'

[22] https://thenextweb.com/apps/2013/10/24/across-desktop-mobile-chrome-used-firefox-ie-opera-combined/

targets are wide. Thanks to this approach, the system's question answering speed is accelerated by 205-253% as it is shown in Table 6.

Table 6. The proposed system's question answering performance through different page sizes

Natural Language Question	Search time for page size is 20 (sec.)	Search time for page size is 1 (sec.)	Performance Gain (%)	SPARQL Query
how tall is Michael Jordan?	128	42	205	SELECT DISTINCT * WHERE { { ?x rdfs:label "Michael Jordan"@en . ?x foaf:isPrimaryTopicOf ?wiki . ?x dbo:wikiPageID ?wikiPageID . ?x dbo:abstract ?abstract . OPTIONAL { ?x dbo:thumbnail ?thumbnail . ?x dbpedia2:name ?name . } FILTER (lang(?abstract) = 'en') } UNION { } } LIMIT 1 OFFSET 0
find places in England	53	15	253	SELECT DISTINCT * WHERE { { ?x dbpedia2:locationCountry :England . ?x dbpedia2:name ?name . ?x dbont:abstract ?abstract . OPTIONAL { ?x dbont:thumbnail ?thumbnail . ?x foaf:isPrimaryTopicOf ?wiki . ?x dbont:wikiPageID ?wikiPageID . } FILTER(lang(?abstract) = 'en') } UNION { ?y dbont:dfE ?x . ?y rdfs:label ?target . ?x rdfs:label ?name . ?x dbont:abstract ?abstract . OPTIONAL { ?x foaf:isPrimaryTopicOf ?wiki . ?x dbont:wikiPageID ?wikiPageID . ?x dbont:thumbnail ?thumbnail . } FILTER(regex(?target, '^England', 'i') && lang(?name) = 'en' && lang(?target) = 'en' && lang(?abstract) = 'en') } } LIMIT 1 OFFSET 0
find people died in 1999	91	27	237	SELECT DISTINCT * WHERE { { ?x dbpedia2:deathDate ?tk . ?x dbpedia2:birthName ?name . ?x dbont:abstract ?abstract . OPTIONAL { ?x foaf:isPrimaryTopicOf ?wiki . ?x dbont:wikiPageID ?wikiPageID . ?x dbont:thumbnail ?thumbnail . } FILTER (regex(str(?tk), '^1999', 'i') && lang(?abstract) = 'en') } UNION { } } LIMIT 1 OFFSET 0

The proposed system is integrated with Freebase and widely used social media platforms in order to give more information about found answers. The proposed system's user interface dynamically changes through types of answers. A home page link, an interactive map and further information from Freebase are shown to users when it is available.

By inspiring from one of the ontologies used in Cinemappy by the Ostuni et al. [16] and presenting information in different languages, we have obtained a faster and more consistent system because of these two ontologies come from the same origin. By using this approach, all limitations based on languages could be eliminated.

When compared to SPARTIQULATION of Ell et al. [17], we use all resource description languages including RDF, RDFS, OWL, FOAF, and even DBpedia's own ontology. With this approach, we get all

the information about resources through DBpedia which is regularly checked and updated by people who is working for it. The proposed system uses only DBpedia as information source while answering questions enabling homogeneity of data and prevents the system from low-quality results.

To prevent from supervising errors or cyber attacks which is possible to occur in systems such as FREyA [20], we offered an unsupervised system that does not need training. It just benefits from users' feedback for their advantages while making recommendations. The system is always consistent and its question answering mechanism is not affected by users. So, our system can be evaluated as more secure and consistent.

Unlike the template queries based system [23] which may be useful to answer questions which exactly fit predefined templates, our system gives some related information about found answers even it is not able to find equal property for the extracted verb. We have used both predefined patterns and instantly created queries through the similarity between DBpedia ontology and extracted natural language questions while preparing SPARQL queries enabling answers to cover as many questions as possible. A comparison of the proposed system with related works is presented in Table 7.

Table 7. The comparison of the proposed system with the related work

Criteria	SPARTIQULATION	PowerAqua	FREyA	WATSON	The Proposed System
RDF, RDFS	+	+	+	+	+
OWL	-	+	+	+	+
Heterogeneous data sources	+	+	+	+	+
Need of training	-	-	+	-	-
Dynamic queries	+	+	+	+	+
Voice search	-	-	-	-	+
Answer vocalization	-	-	-	-	+
Caching mechanism	-	-	-	+	+

5. CONCLUSION

We implemented a natural language question answering system that translates natural languages questions into equivalent SPARQL queries to make it ready to search via DBpedia endpoint service for all users considering all performance issues and availability features. The proposed approach is based on parsing natural language questions, extracting POS tags of questions to identify questions with their answers' types. We have defined some patterns related to these POS tags. The proposed system is very flexible, defined patterns can be easily extended for future works. The query which is constructed through the predefined patterns is merged with the one constructed through the DBpedia ontology after POS tags of natural language questions are extracted. After both SPARQL queries are constructed and merged into one valid query, we query it over DBpedia endpoint service. As we achieved at all, it is the first time a semantic web question answering system offers speech search via voice recognition. Also, answers are vocalized to offer a complete natural language question answering portal for visually-impaired people. The contributions of this study can be summarized as:

- We provide a new approach to process natural language questions using semantic web knowledge bases in order to find the best possible answers to the questions of the end-users.
- The proposed system provides features which are not provided by knowledge bases endpoint services such as paging, dynamic paging, and question-answer caching in order to improve the proposed system's question answering performance.
- To improve the system's usability, voice search, and answer vocalization, social media integration and live spell checking features are provided through the proposed system.
- A recommendation module that uses both semantic and syntactic similarity is developed to provide suggestions for the questions which could not be directly answered by the system.

With all of these approaches, the proposed system is able to find answers to 75% of the most common English question patterns. Also, it is able to find related (*indirect*) to 87.5% of commonly used English question patterns which are listed in Appendix A. Final success is tied to a true match between verb of natural language question and their equivalent property defined on DBpedia. Because there is no exact DBpedia property match with natural language verb, sometimes identical DBpedia property cannot found. For these situations, our system finds the most relevant answers to asked questions.

As for future work, we would like to extend our predefined patterns through a flexible architecture. Another future improvement would be benefiting from multiple knowledge bases with reminding heterogeneity and consistency threats. With utilizing these approaches, we will be able to answer more complex queries than before and give more information from different knowledge bases. Due to Web Speech API is one of the brand-new features of HTML 5 it is currently only supported by Google Chrome/Chromium version 25 and newer versions. As a future goal, we want to implement our own cross-browser speech recognition mechanism in order to overcome this limitation.

For performance improvement, our system appertains to SPARQL endpoint service performance. The only way to improve the system's question answering performance through this service is optimizing constructed SPARQL queries. Therefore, development of a SPARQL optimization tool (API) is one of our future goals. In addition to that, an intelligent caching mechanism which will decide the cache duration through assessing the changeability of the question would improve the performance of the proposed system in term of elapsed time to answer a question.

REFERENCES

[1] Berners-Lee T, Hendler J, Lassila O. The Semantic Web. Sci Am Citeseer 2001; 284(5):34–43.

[2] Metz C. Tim, Lucy, and The Semantic Web. PC Mag 2007.

[3] Rapoza J. SPARQL Will Make the Web Shine. eweek 2006.

[4] Segaran T, Evans C, Taylor J. Programming the Semantic Web. 1st ed. Sebastopol, CA, USA: O'Reilly Media, 2009.

[5] Bizer C, Heath T, Berners-Lee T. Linked data-the story so far. Int J Semant Web Inf Syst 2009; 5 (3):1–22.

[6] Doncel VR. How O is the LOD cloud? 2015. Available at: http://www.cosasbuenas.es/blog/how-o-is-lod-2015; Accessed: 10-Jun-2018.

[7] State of the LOD Cloud. University of Mannheim 2015. Available at: http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/; Accessed: 12-May-2019.

[8] Linked Open Data cloud diagram as of March 2019. Linked Open Cloud 2019. Available at: https://lod-cloud.net/versions/2019-03-29/lod-cloud.png; Accessed: 12-May-2019.

[9] Lehmann J, Schüppel J, Auer S. Discovering Unknown Connections – the DBpedia Relationship Finder. In: Proceedings of the 1st SABRE Conference on Social Semantic Web - CSSW'07; 26-28 September 2007; Leipzig, Germany.

[10] Kabakus AT, Dogdu E. Question Answering System over Linked Data. In: 7th National Software Engineering UYMS'13; 26 September 2013; İzmir, Turkey.

[11] Bizer C, Lehmann J, Kobilarov G, et al. DBpedia - A crystallization point for the Web of Data. Web Semant Sci Serv Agents World Wide Web 2009; 7 (3):154–65.

[12] About | DBpedia. DBpedia 2019. Available at: https://wiki.dbpedia.org/about; Accessed: 12-May-2019.

[13] Why do people use the Internet? addictionblog 2011. Available at: http://internet.addictionblog.org/why-do-people-use-the-internet-10-reasons/; Accessed: 12-May-2019.

[14] Weider K. 15 Reasons Why People Use The Internet. Weider Web Solutions 2016. Available at: https://www.weiderweb.com/15-reasons-why-people-use-the-internet-and-how-to-use-that-to-your-advantage/; Accessed: 12-May-2019.

[15] New AOL Content Research: How Eight Moments Define Behavior Around the World. AOL 2016. Available at: https://advertising.aol.com/en/blog/new-aol-content-research-how-eight-moments-define-behavior-around-world; Accessed: 12-May-2019.

[16] Ostuni VC, Gentile G, Noia T Di, et al. Mobile Movie Recommendations with Linked Data. In: CD-ARES 2013: Availability, Reliability, and Security in Information Systems and HCI; 2-6 September 2013; Regensburg, Germany.

[17] Ell B, Vrandecic D, Simperl E. SPARTIQULATION: Verbalizing SPARQL queries. In: The Semantic Web: ESWC 2012 Satellite Events; May 27-31 2012; Heraklion, Greece.

[18] Lopez V, Fernández M, Motta E, et al. PowerAqua: Supporting users in querying and exploring the Semantic Web. Semant Web 2012; 3 (3):249–65.

[19] Cabrio E, Cojan J, Aprosio AP, et al. QAKiS: An open domain QA system based on relational patterns. In: Proceedings of the ISWC 2012 Posters & Demonstrations Track, volume 914 of CEUR Workshop Proceedings; 11-15 November 2012; Boston, MA, USA.

[20] Damljanovic D, Agatonovic M, Cunningham H. FREyA: an Interactive Way of Querying Linked Data using Natural Language. In: Proceedings of the Workshop on Question Answering Over Linked Data (QALD); 8-9 October 2011; Heraklion, Greece.

[21] Ferrucci D, Brown E, Chu-Carroll J, et al. Building Watson: An Overview of the DeepQA Project. AI Magazine 2010; 31 (3):59–79.

[22] Watson – A System Designed for Answers. IBM Syst Technol 2011.

[23] Unger C, Bühmann L. Template-based question answering over RDF data. In: Proceedings of the 21st international conference on World Wide Web; 16-20 April 2012; Lyon, France.

[24] Unger C, Cimiano P. Pythia: Compositional Meaning Construction for Ontology-Based Question Answering on the Semantic Web. In: Munoz, Rafael, Montoyo, Andres, Metais, Elisabeth, editors. Natural Language Processing and Information Systems. Berlin, Heidelberg, Germany: Springer International Publishing, 2011. pp. 153–160.

[25] Cimiano P, Haase P, Heizmann J, et al. Towards portable natural language interfaces to knowledge bases - The case of the ORAKEL system. Data Knowl Eng 2008; 65 (2):325–54.

[26] Unger C, Freitas A, Cimiano P. An Introduction to Question Answering over Linked Data. In: Reasoning Web 2014: Reasoning Web. Reasoning on the Web in the Big Data Era; 8-13 September 2014; Athens, Greece.

[27] Yahya M, Berberich K, Elbassuoni S, et al. Natural language questions for the web of data. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL'12); 12-14 July 2012; Jeju Island, Korea.

[28] Tran T, Wang H, Rudolph S, et al. Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In: 2009 IEEE 25th International Conference on Data Engineering; 29 March-2 April 2009; Shanghai, China.

[29] Shekarpour S, Ngonga Ngomo A-C, Auer S. Question answering on interlinked data. In: Proceedings of the 22nd international conference on World Wide Web - WWW '13; 13-17 May 2013; Rio de Janeiro, Brazil.

[30] Schmid H. Penn Treebank P.O.S. Tags. University of Stuttgart 2003. Available at: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html; Accessed: 12-May-2019.

[31] Leavitt N. Will NoSQL Databases Live Up to Their Promise? Computer (Long Beach Calif) 2010; 43 (2):12–14.

[32] Okman L, Gal-Oz N, Gonen Y, et al. Security Issues in NoSQL Databases. In: 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications; 16-18 November 2011; Changsha, China.

[33] Hasan SS, Isaac RK. An integrated approach of MAS-CommonKADS, Model–View–Controller and web application optimization strategies for web-based expert system development. Expert Syst Appl 2011; 38 (1):417–28.

[34] Varma V. Software Architecture: A Case Based Approach. New Delhi, India: Pearson Education India, 2009.

[35] One minute social media infographic. Social Jumpstart 2012. Available at: http://socialmediachimps.com/wp-content/uploads/2012/03/one-minute-social-media-infographic.jpg; Accessed: 12-May-2019.

[36] Infographic: How often people interact online. Go-Globe 2013. Available at: https://techmarketingbuffalo.com/wp-content/uploads/2013/06/infographic-how-often-people-interact-online.jpg; Accessed: 12-May-2019.

[37] Wickre K. Celebrating #Twitter7. Twitter 2013[Online] 2013 [cited 2019]. Available at: https://blog.twitter.com/official/en_us/a/2013/celebrating-twitter7.html; Accessed: 12-May-2019.

APPENDIX

Appendix A – Commonly used English question patterns

Question Pattern	Direct Answer	Indirect Answer
What	+	+
When	+	+
Where	+	+
Who	+	+
How …	-	+
Which …	-	-
Whom	+	+