The Knowledge Engineering Review (2022), **37**, e2, pp. 1–42
doi:10.1017/S0269888921000138

The Knowledge Engineering Review
CAMBRIDGE UNIVERSITY PRESS

REVIEW

A survey on semantic question answering systems

Christina Antoniou and Nick Bassiliades

School of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
E-mails: antoniouc@csd.auth.gr, nbassili@csd.auth.gr

Received: 11 December 2020; **Revised:** 25 October 2021; **Accepted:** 27 October 2021

Abstract

Recently, many question answering systems that derive answers from linked data repositories have been developed. The purpose of this survey is to identify the common features and approaches of the semantic question answering (SQA) systems, although many different and prototype systems have been designed. The SQA systems use a formal query language like SPARQL and knowledge of a specific vocabulary. This paper analyses different frameworks, architectures, or systems that perform SQA and classifies SQA systems based on different criteria.

1. Introduction

Recently, serious efforts have been made to organize both general and specialized knowledge in the form of Resource Description Framework (RDF) knowledge bases (KBs) (Mazzeo & Zaniolo, 2016a). More and more structured knowledge is released on the Web (Zou *et al*., 2014). Examples of such KBs are DBpedia[1], Freebase, and YAGO[2]. KBs are used, among others, for text summarization, opinion mining, classification, semantic search, and question answering (Atzori *et al*., 2016). There are two categories of RDF KBs. In the first category, the datasets contain a variety of knowledge in different domains and that is the reason why they are so popular. For example, DBpedia encodes encyclopedic knowledge extracted from Wikipedia. On the other hand, there are datasets that specialize in a single or a closed domain such as Music (MusicBrainz[3], Music Ontology[4], Biomedical datasets (such as SIDER[5], Diseasome[6], Drugbank[7]), or Medicine Ontology[8] and Geography (LinkedGeoDat[9]).

The RDF data model and the SPARQL query language are the standards for modeling and querying data in the semantic Web[10]. The standard format for a KB in the Web is RDF stored in a repository as a set of triples, denoted as <subject, predicate, object>, which is called the RDF graph; in this graph, subjects, and objects are vertices, and predicates are edge labels. The standard way to access RDF data is the SPARQL query language. The syntax and the semantics of SPARQL query language for RDF are determined by RDF specification. SPARQL allows users (experts) to write queries across data, which can be stored according to RDF specification or are created as RDF via middleware. The SPARQL

[1] http://dbpedia.org.
[2] https://yago-knowledge.org.
[3] http://musicbrainz.org.
[4] http://musicontology.com.
[5] http://sideeffects.embl.de.
[6] http://wifo5-03.informatik.uni-mannheim.de/diseasome/.
[7] http://www.drugbank.ca.
[8] https://bioportal.bioontology.org/ontologies/OGMS.
[9] http://linkedgeodata.org/About.
[10] http://www.w3.org/.

Cite this article: C. Antoniou and N. Bassiliades. A survey on semantic question answering systems. *The Knowledge Engineering Review* 37(e2): 1–42. https://doi.org/10.1017/S0269888921000138

© The Author(s), 2022. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution and reproduction in any medium, provided the original work is properly cited.

syntax and the need to understand the RDF Schema for the RDF datasets make it difficult for common users to search and find relevant information (Zou *et al*., 2014).

Question answering (QA) systems give answers in response to questions in natural language. Question answering is not only an area from information retrieval (IR), but it is an area in the field of information extraction (IE) and natural language processing (NLP) as well (Allam & Haggag, 2012; Sasikumar & Sindhu, 2014). The three basic components of question answering are as follows: question classification, IR, and answer extraction (Allam & Haggag, 2012). According to Bouziane *et al*. (2015), the three basic components are as follows: Question Analysis (included classification and extraction, extended keywords, and Named Entity Recognition), Document Retrieval, and Answer Extraction which are included in most QA systems. However, these components may have differences due to their implementation of every component.

In Semantic Web, educated users can use SPARQL and can thus search for information in the KBs. On the other hand, ordinary users that are not familiar with SPARQL face difficulties in searching linked datasets. A solution to this problem is semantic question answering (SQA) systems. In these systems, users ask questions in natural language using their own terminology and receive a response generated by searching in an RDF KB. Through semantic question answering (SQA) systems, users overcome two major barriers: using a dedicated query language, such as SPARQL, and having a perfect knowledge of the KB specific vocabulary (Höffner *et al*., 2017).

The research on question answering systems using linked data is very active, and there is a wide variety of methodologies. Many prototype SQA systems have been developed for different datasets. The users of such systems may ask questions in natural language. However, because of the complexity and ambiguity of natural language, the systems need to have many different steps for achieving the task of understanding users' informational needs. So far, there is a survey in Höffner *et al*. (2017) that describes challenges and solutions for SQA systems. Also, a survey by Diefenbach *et al*. (2018a) referred to techniques and steps used in SQA systems. This article distinguishes categories of SQA systems based on criteria in order to lay the groundwork for a collection of common practices as no categories of SQA systems have been identified. This categorization can also serve as an archive of frameworks and systems where each system is classified according to the techniques that it uses for various criteria, such as types of questions, types of analysis done on questions, types of representations used for questions, and their matching functions. This can help developers, or anyone interested to find out directly the technique or steps used by each system, or to benchmark her own system against existing ones.

The rest of the paper is organized as follows. In Section 2, we first briefly provide related work. In Section 3, we present and discuss the classification of semantic question systems based on the literature surveyed of QASs. We were also inspired by this classification and by the criteria in QASs of Mishra and Jain (2016). The classification is based on types of domains, types of data sources, types of questions, types of analysis done on questions, types of matching functions, characteristics of the KB, interaction of user, and answers. We describe SQA systems in Section 4. The systems, given as an example in Section 3, are presented in more detail in this section. Finally, we present the conclusions in Section 5.

2. Related work

There have been quite few surveys on question answering systems and even less in SQA systems. Below, we briefly present the main features that each one of them is focused on and then we state how our survey differs.

Sasikumar and Sindhu (2014) present various methods of Natural Language Question Answering System (NLQA) for general languages. Kolomiyets and Moens (2011) give an overview of question answering technology from an IR perspective. The authors provide a general architecture of question answering where the complexity of the representation level of questions and information objects is increased. Allam and Haggag (2012) describe an overview of question answering and corresponding

systems' architecture, and they also analyze the proposed QA models discussing their advantages, disadvantages, and the results of relevant experiments.

Bouziane *et al*. (2015) refer various QASs and present statistics and analysis of these systems. The authors mention that their survey will help researchers to find the best solution to their issue. Also, researchers can perceive the disadvantages and create new systems for complex queries or even adapt or reuse QASs techniques.

Mishra and Jain (2016) classify question answering systems (QASs) based on different criteria, such as application domain, types of questions dealt, types of analysis done on questions, types of data sources, types of matching functions, and characteristics of data sources, techniques used in QASs, and forms of answer generated. Also, they determine for each category their status (until that date) and make suggestions for future research. Fader (2014) makes a detailed study and extended overview of the open question systems. The author distinguishes two important challenges. The first concerns how systems represent knowledge in order to provide an answer. And the second challenge is related to how the system matches the questions of the user with the queries on the KB. There is a report for open question answering over curated and extracted KBs for the specific challenges, which, however, does not specialize in SQA systems, on RDF triplets and SPARQL queries. Höffner *et al*. (2017) initially give an overview of QA and SQA systems. After reviewing the state of the art for SQASs and for QASs, the authors present some of the systems. They then describe the challenges faced by SQA systems, identify different techniques of each challenge, and propose guidances for the systems that will be created in the future.

Diefenbach *et al*. (2018a) provide an overview of the techniques mentioned in the recent question answering system over a KB. Their purpose is to group techniques that implement the same task and describe the advantages and disadvantages of each technique.

Dimitrakis *et al*. (2020) make a review of the question answering systems over Linked Data, documents, and hybrid data sources (Linked Data and documents). They also describe the steps used in the question answering process but also methods and techniques for implementing these steps. They distinguish some criteria (knowledge source, types of questions, and domain type) and categorize the question answering systems according to them. Finally, they categorize evaluation and training datasets according to some criteria, such as domain, tasks, evaluation metrics, and knowledge source.

Regarding surveys on SQA systems, we observe that Dimitrakis *et al*. (2020) and Bouziane *et al*. (2015) generally refer to question answering systems. *Dimitrakis et al*. (2020) provide the main categories of question answering systems and do not specialize in SQA systems, and they cite 3 categories for question answering systems in general. On the other hand, Höffner *et al*. (2017) describe the challenges such as lexical gap, ambiguity, multilingualism, complex queries, distributed knowledge, procedural, temporal and spatial questions, and templates faced by these SQA systems. Additionally, they depict solutions and provide recommendations for future systems. Also, the article of Diefenbach *et al*. (2018a) similarly reports techniques and steps used in SQA systems such as question analysis: recognizing named entities, POS (Part Of Speech) tagging, dependency analysis, phrase mapping: string similarity, semantic similarity, disambiguation: Hidden Markov Model, Integer Linear Program, query construction: semantic parsing, templates. The majority of techniques presented in the works of Höffner *et al*. (2017) and Diefenbach *et al*. (2018a) are similar.

Compared to the previous surveys, our survey differs in the classification of SQA systems based on different criteria and the effort to identify common features of SQA systems. To date, there are no surveys for the categorization of SQA systems. This is the innovation and contribution of the current paper. We have augmented the criteria from the criteria of QASs of Mishra and Jain (2016), by adding and supplementing more appropriate criteria according to the study of SQA systems that we have conducted. Summarizing, the differences in our classification criteria with those of Mishra and Jain (2016) can be grouped into 3 categories:

1. differentiated criteria (in terms of the values they get), which are indicated with an asterisk in Table 1,

Table 1. *Criteria for classifying SQA systems.*

Criteria	Values	
Types of domains	Open domain Closed domain **Hybrid domain**	
Types of data sources	Structured data sources Structured and Unstructured data sources (hybrid data sources)	
Types of questions	Wh-question type question or factoid type questions or factual questions List Boolean Count*	
Types of analysis done on questions	Morphological Syntactical **Dependency** Semantic Expected answer type Focus recognition of questions Pragmatic and discourse (in hybrid SQASs)	
Types of representations used for questions and their matching functions	Algebraic models or feature-based models* Probability models Semantic graph-based models* Theoretic models (in hybrid SQASs)*	
Characteristics of the KB	**Multilingualism** **Distributed knowledge base**	
Techniques in SQASs	Machine learning* Natural language processing Information retrieval Knowledge retrieval and discovery	
Interaction of user	**User Interaction in disambiguation** **Type of users**	
Answers	Extracted answer	Date (if there is one registered in KB otherwise as generated)* Resource* String*
	Generated answer	Boolean Numeral*

The '*' means that a criterion, sub-criterion or value that is different from its counterpart at Mishra and Jain (2016). Boldface indicates a new criterion, sub-criterion or value, in relations to Mishra and Jain (2016).

2. criteria not included in Mishra and Jain (2016) but included in our survey because they are used in SQAs, which are indicated with italic and boldface font in Table 1, and
3. criteria included in Mishra and Jain (2016) that have been replaced in our survey with other, more appropriate, criteria, indicated with boldface in Table 1.

In this paper, we analyze SQA systems, some of which participated in the QALD competitions or were evaluated with benchmarks. Our main difference with previous similar surveys, such as Höffner *et al*. (2017), is that we describe, analyze, and classify several SQA systems, while Höffner *et al*. focus on the detailed description of the various components of the SQA systems. Also, we distinguish and record the common methods or practices or architectures that these systems use. These systems have been evaluated and compared quantitatively with other systems through the QALD competitions or benchmarks. For this reason, these systems were selected, so that their performance in similar tasks could be comparable. In addition, we summarize the architectures of these systems to enable the reader to study them deeply and choose the one that best suits their direction.

3. Criteria for classifying semantic question answering systems

In this section, we discuss details of the proposed classification of SQA systems. A description of the classification will be given. Also, we provide a detailed description of the systems for each category or class so that researchers can see the pros and cons and suggest solutions for complex SPARQL queries. The systems, given as an example, are presented in more detail in the next section in which there are also their references. The readers or researchers can also reuse or adapt SQA techniques. Finally, we compare the criteria of the QASs of Mishra and Jain (2016) with those we created for the SQA systems. We identify criteria for classifying SQA systems based on the literature survey of QASs.We were also inspired by this classification and by the criteria in QASs of Mishra and Jain (2016). The classification is based on types of domains, types of data sources, types of questions, types of analysis done on questions, types of representations used for questions and their matching functions, characteristics of the KB, interaction of user, and answers. Table 1 briefly describes the criteria for classifying SQA systems.

3.1. Classification based on types of domains: open, closed, or hybrid domain

According to Mishra and Jain (2016), the above category for QASs is divided into general domain and restricted. Questions in open domain or general domain are general. Mishra and Jain (2016) report (for question answering systems in open domain) that the number of ordinary users in open domain is greater and open domains are more suitable for ordinary users. Open domain users' questions do not require special knowledge (special vocabulary or special keywords). Another advantage of open domain is that their repositories are large. The characteristics of QASs in closed domain or restricted domain are the high quality of the answers due to the restricted domain. The users are usually experts to the specific domain and the content of the questions is based on the vocabulary of the closed domain (Mishra & Jain, 2016). We have differentiated this criterion in our survey by also adding hybrid domain because there are systems that accept questions for both open and closed domains.

3.1.1. Open domain SQA systems

Questions in open domain or general domain are general. Mishra and Jain (2016) report (for The above advantages in general domain of QASs apply to SQA systems in open domain. English DBpedia contains 3.5 million entities, which is extracted from Wikipedia, 320 classes, and 1650 properties. DBpedia version 3.6 consists of approximately 280 million RDF triples and 3.7 consists of approximately 370 million RDF triples. DBpedia also contains links to YAGO categories. Of course, open domain KBs can be large but they have several disadvantages such as ambiguity, imperfection, incomplete domain and range for properties, unspecified entity types, complex semantic entity labels, redundancy in properties, errors in the range of properties so-called modeling errors. An important drawback is that the quality of responses is characterized as low compared to closed domains. This is because closed domains are associated with an unambiguous ontology while open domains are associated with an ambiguous ontology (Lopez *et al*., 2013).

Examples of SQA systems using KB open domain are as follows: Swipe, Casia@V2, Deanna, SMIQ, YagoQA, YodaQA, Qanswer, gAnswer, ISOFT, SQUALL2SPARQL, Scalewelis, Intui2, Intui3, OAKIS, SemSek, Alexandria, RTV, Sorokin and Gurevych (2017), Athreya *et al*. (2021) and Liang *et al*. (2021). The most widely used and most popular open domain KB is DBpedia. The QALD competition also evaluates the systems using Dbpedia as open domain and Musicbrainz as closed domain. Lopez *et al*. (2013) report that DBpedia is considered 'the central interlinking hub for the emerging linked data cloud'. Other open domain KBs are Yago and Freebase.

3.1.2. Closed domain SQA systems

The characteristics of closed domain in SQA systems are similar to those of QA systems. The content of closed domain SQASs systems varies: geography, music, medicine, and biomedicine. Examples of SQA systems that use closed domain KB are as follows: GFMed, POMELO, and Swip. Of course, SWIP was evaluated in open domain; however, Swip cannot be characterized as a hybrid as the results on the DBpedia dataset were not encouraging. GFMed and POMELO query biomedical linked data such as Diseasome, SIDER, and DrugBank. The QALD competition also evaluates the systems using MusicBrainz as closed domain. Other closed domain KBs are DrugBank, Diseasome, SIDER, which are derived from biology field, and Mooney Geoquery[11] (dataset).

3.1.3. Hybrid domain SQA systems

There is a third category, the hybrid domain. This category does not exist in the QASs of Mishra and Jain (2016) but many SQA systems are able to use open domain and closed domain. QALD competition also evaluates the systems in open domain and closed domain; the datasets that are selected are DBpedia and MusicBrainz. Some of these systems that are characterized as hybrid ones are CANaLI and Freya. CANaLI has access to specific domains such as biology (DrugBank, Diseasome, and SIDER) and music domain (MusicBrainz). This system uses DBpedia as well. Freya system uses Mooney Geoquery and MusicBrainz as closed domains and DBpedia as open domain. WDAqua-core1 (Diefenbach *et al*., 2018b, 2020) can support several open domain KBs such as Wikidata, Dbpedia, and Freebase and closed domain KBs such as MusicBrainz and DBLP at the same time.

3.2. Classification based on types of data sources

According to Mishra and Jain (2016), the above category for QASs is divided into structured, semi-structured, and unstructured data sources. MySql, SQLite, and DB2 are considered as structured data. The characteristics of structured data are as follows: they are identified as entities that are related to each other with relationships, have a predefined form and schema, have query language, there is a direct correlation of query with structured data, the answers are considered very reliable, the structured data are not subject to complex NLP, the creation of structured data sources is a laborious process, the information stored in data structured sources is limited, and for each structured data source, there is a specific representation, and therefore, the question conversion is based on the formula of data sources. Regarding the characteristics of semi-structured data (such as markup languages, e-mail, and EDI) they are the following: schema defines the representation of information, it is flexible as data exchange between different types of databases is allowed, structured data is converted to semi-structured data, creating semi-structured data is a time-consuming process. Finally, unstructured data sources have different types, their representation does not have a specific format as in the previous data source, adding and updating information is easy, unstructured documents require NLP and IR technologies; also, the answers are characterized of low degree of reliability.

SQA systems are divided into structured data and hybrid (structured data and unstructured data). According to the literature review and study of different and prototype systems, SQA systems always

[11] https://www.cs.utexas.edu/users/ml/nldata.html.

look for answers in structured data (linked data) and some, less in number, use structured data and unstructured data (documents), which are called hybrid. As the name of the systems (SQA) refers, these systems were created to look for answers in linked data KB.

3.2.1. Structured data sources

The majority of SQA seek answers to users' questions in open domain KB (over linked data) such as DBpedia, Yago, and Freebase. But some of them are looking for answers in closed domains such as MusicBrainz, DrugBank, Diseasome and SIDER and Mooney Geoquery. And some in both open domain KB and closed domain KB as described above. Stored data in KB follow a strict structure and have a schema, they have a query language, the creation of KB is a time-consuming and demanding process. The systems accept users' questions and convert them into SPARQL queries to search for answers in KB. Usually, in order to make the conversion, the question elements must be matched with the KB elements to create the SPARQL query, the so-called matching.

3.2.2. Structured and Unstructured data sources (hybrid data sources)

There are systems such as ISOFT and YodaQA that utilize unstructured documents, which are combined with linked data to generate answers to users' questions using both types of data sources. KB responses are more accurate than systems that use IR (Park *et al*., 2015; Mishra & Jain, 2016).

For example, ISOFT combines IRQA (Information Retrieval Question Answering) approach and KBQA (Knowledge Base Question Answering) approach. This system developed a multi-information tagged text database based on Wikipedia using co-reference resolution and disambiguation methods. ISOFT usually divides questions (when they are large) into sub-queries, and for each sub-query, it looks for the answer in a multi-information tagged text database. The process continues until system finds the final answer. If it does not find the answer in the multi-information tagged text database, then it creates the corresponding SPARQL query and looks for the answer in DBpedia.

YodaQA accepts the user's question in natural language and discovers the answer based on both unstructured (English Wikipedia, enwiki) and structured KB (Dbpedia, Freebase). It is worth mentioning that it uses more than one KB. Here, we observe a split of the question into features for which answers are sought in unstructured and structured data sources. The responses that are discovered by hybrid data sources are merged and sorted. Un-structured documents require NLP (POS, stemming, parsing, dependency parsing, tokenization), disambiguation methods, IR and information extraction, named entity recognition, co-resolution, and other approaches related to NLP. Note that NLP of both question and unstructured documents is performed when SQA systems use hybrid data sources. On the other hand, in structured data, NLP is applied only to the user's question.

3.3. Classification based on types of questions

Mishra and Jain (2016) divide QASs questions into factoid type questions, list type questions, hypothetical type questions, confirmation questions, causal questions. Factoid type questions include words that start with wh, that is, what, when, which, who, how. Usually, the QASs' performance on these questions is satisfactory. A large repository is available for these types of questions. NLP, applied to specific questions by QASs, is not complicated in order to answer the questions asked. List type questions include as answers lists of entities or events. Named entities are the expected type of answers for list type questions. Their performance is quite good. As with factoid type questions, no complex NLP is required. List type questions generally have difficulty with the threshold value, which determines the quantity of entities. Also, the techniques of factoid type questions are equally satisfactory in the list type questions. Hypothetical type questions do not have a specific expected type of answer and that is why the accuracy of the answers is characterized by a low degree as well as the reliability. Even the techniques of factoid type questions do not work for hypothetical type questions. Causal questions such as how

and why ask for explanations for an entity. A sentence, a paragraph, or an entire text can be included as answers. Causal questions require complex NLP. The answers to the confirmation questions are yes or no. Approaches such as inference mechanism, world knowledge, and common-sense reasoning are needed to extract answers to confirmation questions. They also mention opinion questions. These questions use the social Web and opinion mining techniques to find the answers. We divide SQASs questions into factoid, list, boolean, and count. Causal question type in OAs can apply only in hybrid (structured and unstructured) SQA systems.

From the study and the literature review of SQA systems but also from the questions (Zou *et al*., 2014; Tran & Nguyen, 2016) of the annual competitions that are held, we observe that the types of questions are the following: factoid, list, boolean (confirmation questions in QA), and count question. Count question type does not exist in QAs but other types, such as factoid, list, boolean, do. The first category is wh-question type question or factoid type questions or factual questions. These questions usually start with the word wh-, such as which, when, who, what, where or in which, and how are answered by an entity (person, organization, location, date-day-date, etc.). They usually consist of most of the questions.

The second category is list type questions, which start the question with the following words 'Give me all or similar phrases'. The expected answer type is the same as in the factoid type question, but the difference is in the number of answers. For example, the following factoid type question asks the capital with the largest population 'What is the capital with the largest population?' On the other hand, a list type question could be 'Give me all capital which have a population greater than 10 000 000?' In both cases, the expected type is the same (capital) but usually the users, who type list type questions, wait for more than one answer of the same expected type. Note that the following question 'In which countries can you pay using the West African CFA franc?' can be considered a list type question.

The third category is called boolean type question and is also called dichotomous (Pradel *et al*., 2012) type question. Users when entering a boolean type question expect an answer yes or no, true, or false. These questions usually start with an auxiliary verb.

There is another kind of question that is observed in SQA systems that is often referred to as count question (Pradel *et al*., 2012; Shizhu *et al*., 2014; Beaumont *et al*., 2015). It could be considered as a separate category because several systems (Pradel *et al*., 2012; Shizhu *et al*., 2014; Beaumont *et al*., 2015) refer to it as a separate type of question. In the annual competitions, it is observed as a separate type of question.

Several systems (such as CANaLI, Intui2, IQA, LAMA, ComQA, GFMed) distinguish questions into simple and complex questions. EARL (Entity and Relation Linker) (Dubey *et al*., 2018.) separate the questions into simple and complex ones. The questions, that create a triple, are considered as simple question. Examples of such questions are 'Who is the spouse of Barack Obama?' The four states (including initial and final state) of the CANaLI finite automaton can support a wide range of factual queries. Question like 'Who are the spouses of politicians having birthplace equal to United States?' It is considered more complex than the first question about CANaLI, but it is classified as simple questions. Questions like 'Give me the cities having population greater than that of Los Angeles', 'Give me the country having the 2nd largest population' (Atzori *et al*., 2016; Mazzeo & Zaniolo, 2016a, 2016b). They are referred to as complex questions (complex does not refer to the list type but to the creation of the complex query). On the other hand, Swipe cannot accept questions in natural language however the user can formulate questions based on Search by Example. The types of questions observed are list and count type. The creators of Swipe identify as examples of complex questions the following: 'Who are the US presidents who took office when they were 55-year-old or younger, during the last 60 years', 'Find the town in California with less than 10 thousand people', 'What is the average population of California cities with less than 10 thousand people', and 'What is the largest of those cities and its population?' This system supports complex queries (i.e. those that require aggregates, joins) and historical queries (Atzori & Zaniolo, 2012; Atzori *et al*., 2016).

Intui2 system can answer only simple questions. QAKiS can handle simple queries (one triplet). Intui2 system was evaluated in QALD-3 DBpedia test and QALD-3 DBpedia train. LAMA, ComQA, and IQA system can support complex questions (multiple entities and relations). LAMA handles the following types of questions: factual, list, boolean, and count. SemSek cannot support complex queries as SPARQL aggregation and ask type queries. Intui3 (Dima, 2014) can deal with list type question, factual type question, count type question but not boolean questions.

Xser cannot handle temporal information. It was also evaluated in 50 test questions (includes list, factual, boolean, count type question) of QALD-4. The creators of CASIA@V2 divide the questions into boolean, number (count question), and normal (list type and factual type question). Xser, CASIA@V2, SMIQ, GFMED, and POMELO were evaluated in the QALD-4 test dataset (includes list, factual, boolean, count type question). SemGraphQA identifies three types of count question, boolean, and factual question. QAnswer and SemGraphQA were also evaluated in a test questions set (includes list, factual, boolean, count type question) of QALD-5. The system gAnswer was evaluated in a test questions set (includes list, factual, boolean, count type question) of QALD-3. ISOFT develops a task that detects the question type: boolean, factual, and list type or questions that include arithmetic information (comparative, superlative) and simple questions (do not include boolean and arithmetic information questions). It creates a SPARQL query for one triple. Detection task helps to create the appropriate SPARQL template. SWIP detects, factual, list, boolean, and count. Sorokin and Gurevych (2017), Athreya *et al*. (2021) and Liang *et al*. (2021) were evaluated in a test question set of QALD-7 that includes list, factual, boolean, count type questions.

3.4. Classification based on types of analysis done on questions

Mishra and Jain (2016) present the following categories for the analysis of questions: morphological analysis, syntactical analysis, semantic analysis, pragmatic and discourse analysis, expected answer type analysis, and focus recognition of questions. Morphological analysis is responsible for detecting morpheme and assigning a class to each morpheme. This form of analysis is performed using stemming and lemmatization of words. This type of analysis supports in effective way the search. On the other hand, stemming can lead to erroneous results. Regarding syntactical analysis, the result of this analysis is a parse tree, in which each word is identified grammatically. This analysis helps for effective search. Its disadvantage is that it can lead to syntactical ambiguity. The aim of Semantic analysis is to extract the meaning of the question based on the parse tree of syntactical analysis. QASs apply Semantic analysis (such as semantic role labeling) at lexical and sentence level; also, this analysis is not recorded at document level. Pragmatic and discourse analysis is performed at sentence or higher level. This analysis aims to determine connections or relationships (discourse) between the sentences of a text. Opinion, causal, hypothetical, and boolean questions need discourse analysis. Expected answer type analysis determines the expected answer type based on the type question. Expected answer type of factoid type questions and list type questions contribute to the creation of answers. However, this is not the case for causal questions as there is no unique type of answer. Focus recognition of questions helps to create correct answers. We identify the following values for this criterion: morphological analysis, syntactical analysis, dependency analysis, semantic analysis, pragmatic and discourse analysis, expected answer type analysis, and focus recognition of questions. That is, we have added one more value, compared to Mishra and Jain (2016), that of dependency analysis.

A type of question analysis that is very common in SQA systems is semantic analysis because the linked data are in RDF form. Semantic analysis occurs in almost all systems except for the Swipe system. This system uses Search by Example and directly produces a SPARQL query. Dependency analysis is used quite often, less than semantic analysis. This is because dependency analysis determines the relationships of the question elements to each other, and since the stored linked data are in the form of semantic graphs, the two types of analysis are very close to each other. Before presenting the type of analysis in questions for the SQA systems, it is mentioned that in order to find the answer, the following is required: analysis of the question and creation of SPARQL query. In many systems, we also have an

intermediate stage of matching the question with KB. These are the basic stages of SQA systems. Each system implements the above steps with its own implementation method. This is the reason why there is so much originality in these systems. Also, in some systems the mapping of the data with the KB is done during the analysis of the question (such as CASIA@V2).

Xser: analyzes the user question. First, detects the semantic item of the question with a phrase detector using a structured perceptron. Then using a semantic parser (dependency parser), a transition-based DAG parsing algorithm, extracts the triplets in the question.

CASIA@V2: includes detection of the semantic items of the question using rules, it matches the detected phrases with data from KB, and it extracts features using the dependency parse tree. The above is done in order to create semantic triplets and consequently a graph. Later, it resolves ambiguities.

ComQA: detects relations with the use of Stanford Parser which creates the dependency tree. It also discovers the syntactic structure and the hidden structure of the dependency tree to create RDF triples. Then, the system creates query graphs by joining the RDF triples.

DENNA: initially detects the semantic items such as relations, entities, and classes of the user query using detectors. Each detector is responsible for discovering a specific category of semantic data. Detectors use different named entity recognizers and different techniques to determine relationships. All detectors operate independently; this leads to overlapping and ambiguity problems. It then matches the phrases with elements from the KB using dictionaries. It is possible to identify many candidate matches for a phrase. The system then synthesizes the phrases to create triplets (subject property object) using the dependency parser. The ultimate goal of the system is to extract a subgraph after resolving ambiguities.

SMIQ: first detects the proper nouns or name entities so that during parsing which will be done in the next step they are not lost. Then, SMIQ performs dependency analysis taking into account the proper nouns. Inference rules are applied to the set of dependencies to create the semantic interpretation of the question, which is called SMIQ. This structure (SMIQ), which is in the form of triplets, allows the user question to be linked to the KB elements.

PowerAqua: performs a linguistic analysis to convert the question into triplets with interdependencies and then marks a label (subject, predicate, or object) on each term of the triplet. Then to match the question elements with the data from the KBs, the system applies syntactic techniques.

SemGraphQA: detects entity, type, and relations, these are accompanied by a score, and the dependency graph is created. Also, the system applies syntactic analysis. Then, for each word of the dependency graph is determined as the semantic nature (entity or relation). Because the ambiguities are maintained, many graphs are created. This semantic question answer system uses expected answer type analysis.

QAnswer: first generates the dependency tree and then detects DBpedia individuals, types, and properties in the question. Different methods are applied and developed to detect each resource. A large number of graphs have been created following the detection steps. Each graph must match the KB elements. Also, the system applies expected answer type analysis and morphological analysis.

gAnswer: first a dependency tree is created from the question, the relations in the tree are extracted. Then, the semantic relations are connected, and a semantic query graph is created. Subsequently, the graph is mapped to the elements of the KB. At this point, all ambiguities are maintained. It is possible to find many subgraphs, and each one has a score.

OAKIS: after the system identifies the named entity and the expected answer type,it creates the typed question and assigns relational patterns (triplets). Specifically, for each typed question a set of patterns are retrieved. Also, the system applies morphological analysis.

SemSek: performs dependency and syntactic analysis. It applies name entity recognition. The system identifies DBpedia instances and classes on the graph (semantic item detection). System matches the question with KB data and resolves ambiguities.

YodaQA: pos-tagging, dependency parsing, and name entity recognition are applied to the user question. Also, this SQA system uses expected answer type analysis and focus recognition of questions.

ISOFT: the system figures out tokenization, part of speech tagging, dependency parsing, keyword extraction, term extraction, and named entity (NE) extraction, semantic answer type, lexical answer

type, Q2S analysis. It is possible for a user question to create more than one query which means that the question must be split. The system uses focus recognition of questions.

POMELO: the system applies word segmentation, part of speech tagging and lemmatization of the words, with TreeTagger, identification of semantic entities, extracting term and syntactic parsing. Also, POMELO employs expected answer type analysis and focus recognition of questions.

Intui2: the syntax tree is made from the question. Also, the analysis of the user question includes tokenization, lemmatization, and POS tagging and the expected answer type.

Intui3: performs syntactical and semantic analysis. Also, the analysis of the user question includes tokenization, lemmatization, and POS tagging, name entity recognition and chunking.

FREyA: the system applies syntactic parsing to the question. Then takes the result from the syntactic analysis and combines it with heuristic rules to find possible ontology elements from the question in order to link to the ontology elements. Finally, it applies expected answer type analysis.

The system of Sorokin and Gurevych (Sorokin & Gurevych, 2017): it applies tokenization, POS tagging, and extracting fragments. After a list of Wikidata entities are found for each fragment, the system creates possible semantic representations as graphs based on iterative representation generation. Essentially, it performs phrase mapping elements of the question with the elements of the KB. The candidate semantic graphs and the question are converted into fixed-size vectors based on a CNN-based model.

The system of Athreya *et al*. (2021): dependency parsing and POS tagging are applied. Syntactic parsing is vectorized for the recursive neural network model. A Tree-LSTM is applied to learn the question, and the top-n SPARQL templates are found. The candidate SPARQL templates are filled. The template is mapped to the KB and stands for the semantic representation of user's question.

The system of Liang *et al*. (2021): the system applies tokenization, POS tagging, lemmatization, and dependency parsing. The question type classification is identified using machine learning (ML). Phrase mapping of the elements of question to the elements of the KB is applied. Candidate SPARQL queries are created and are ranked based on a Tree-structured Long Short-Term Memory (Tree-LSTM).

SWIP: the name entities are identified. Then, the dependency parser extracts the dependency tree of the user question, it has taken into account the name entities and subsequently detects what the question is asking (focus recognition).

RTV: initially, the system applies dependency parsing to the question and the dependency tree is constructed. Also, the system applies syntactical analysis. The ontological elements of the tree are detected to create the Markov chain and match with the KB elements.

Below we present the question analysis for SQA with CNL. The analysis of question in the CANaLI system is done in a different way. First, the language accepted by the CANaLI system is controlled language; however, the processing of the string (that the user enters) is analyzed using a finite automaton. At the same time as the user enters the system sends a query to the Lucene Index, which sends acceptable tokens according to the function of the finite automaton. These must (a) be semantically correct according to KB (semantic analysis), (b) have a type according to the current and previous state (i.e. be syntactically correct), and (c) be a phrase that matches the string. SQUALL2SPRAQL uses controlled natural language SQUALL. The syntactic and semantic analyses of SQUALL are implemented as a Montague grammar. Scalewelis accepts a CNL and applies syntactic, semantic analysis, and focus recognition. The GFmed system applies controlled natural language and uses a Grammatical Framework (GF). The GF is suitable for multilingualism and divides into abstract and concrete grammars (one for the natural language and one for the SPARQL queries).

In SQA systems, we observe that most of them use semantic analysis due to the linked data in KB. Also, another type of analysis that is often used in SQA is dependency analysis. Dependency analysis is used oftenly. This is because dependency analysis determines the relationships of the question elements to each other, and since the stored linked data are in the form of semantic graphs, the two types of analysis are very close to each other. This type of analysis does not exist in QASs. We also identify morphological analysis, syntactical analysis, semantic analysis, expected answer type analysis, and focus recognition of questions. Pragmatic and discourse analysis do not exist in SQA systems. But this type of analysis

can be used in hybrid systems, which use structured and unstructured data sources (hybrid data sources) to generate the user response. More specifically, pragmatic and discourse analysis can be performed on unstructured data sources. Semantic analysis is essentially the detection of semantic items and the connection between them. Therefore, the type analyses used in SQA systems are morphological analysis, syntactical analysis, dependency analysis, semantic analysis, expected answer type analysis, and focus recognition of questions. Therefore, in relation to the values of the criterion 'Types of analysis done on questions', we have added the value of dependency analysis.

3.5. Classification based on types of representations used for questions and their matching functions

Mishra and Jain (2016) report the following categories of matching functions: theoretic models, algebraic models, probability models, feature-based models, and conceptual graph-based models. First, matching information is characterized in different retrieval models the way documents are represented. Each retrieval strategy has a specific representation model. SQA systems are characterized as prototypes in terms of their implementation, and we distinguish various matching function which are described below. Because the differences between feature-based and algebraic models are not obvious, we consider that all feature-based models are also algebraic models, and we present them in that way. We consider semantic graph-based models which is a generalization and include RDF graphs instead of conceptual graph-based models. Of course, the answers since they come from RDF database are graph-based, by definition. If we consider that the question is converted from free text to SPARQL then the question also becomes graph-based since the SPARQL questions also identify an abstract graph. All SQA system use semantic graph-based models. Our values for this criterion are algebraic models, probability models, semantic graph-based models, and theoretic models.

CANaLI works as a finite automaton. When the user enters the string S at the same time, the Lucene index returns the results matching the string S. The returned tokens must be syntactically correct according to the grammar of the language, semantically correct according to KB, and there must be a phrase that matches the string S. Therefore, the representation of the question functions as a finite automaton.

The operation of Swipe is based on Search by Example (SBE). The user query is represented as the SBE conditions (the user requests a Wikipedia page and fills in the conditions), and the SPARQL query is created. The Query Manager accepts as pairs (field-ID, condition) the user's conditions, which are converted to equivalent DBpedia property names and generate the SPARQL query.

Xser represents the user question as a graph specifically trying to create triplets and associate these triplets. The mapping for subject and object lies in probabilities using the Freebase search API while for predicate is based on the Naive Bayes' model for calculating probabilities. The resolution of ambiguities is based on structured perceptron.

CASIA@V2 creates a graph (of course the graph is created in the pre-last phase of the system in contrast to Xser where from the first steps of question analysis the dependency tree is created). The matching of entities is done using Wikipedia and not only DBpedia using anchors, redirections. For the classes, the system uses the word2vec tool to calculate the similarity between the elements of the sentence and the KB, after converting the phrases into vector. PATTY and ReVerb are used to map properties. The system detects the relationships of semantic properties in DBpedia with relations patterns of resources (Patty and ReVerb) using instance alignments. Afterward, if the question phrase is matched with a relation pattern then the corresponding DBpedia properties are the candidates. The resolution of ambiguities is based on the joint fashion with a Markov Logic Network (MLN).

DEANNA represents the question in the form of triplets and consequently as a graph. The matching of entities and classes with KB elements is based on a dictionary. The same applies to the matching of relational phrases to semantic relations which is based on a dictionary of textual pattern. The resolve of ambiguities is based on creating or exporting a subgraph from the weighted disambiguation graph using an integer linear program (ILP) and the Gurobi ILP solver.

SMIQ applies dependency analysis to represent the question in the form of triplets and consequently as a graph. The structure (SMIQ) is in the form of triplets and is created using Prolog to extract meanings

from rules and dependencies of the dependency tree. This structure provides the ability to link the user question to the KB elements.

SemGraphQA converts the user question to graph or graphs (if there are ambiguities) using dependency analysis. Each graph corresponds to the elements of the KB, and each graph is assigned a score, which is based on a formula. Graphs are sorted by score and SPARQL queries are created.

QAnswer represents the question in graph using dependency analysis. Each graph must match the elements of KB. Each match is assigned a score based on a formula. The system selects the graph with the highest score and creates the appropriate SPARQL query.

gAnswer converts the user question into a graph. It is possible to find many subgraphs during matching with KB, and each has a score based on confidence probabilities. The system detects top-k subgraphs, and this is NP-hard problem. Each subgraph implies an answer to the query; at this point, the ambiguities have been resolved.

ComQA converts the user question to graph or graphs using dependency analysis. Initially, it detects relations with the use of the Stanford Parser that creates the dependency tree. Then, it discovers the syntactic structure and hidden structure of the dependency tree to create RDF triples. In the next phase, the system creates query graphs by joining the RDF triples. In the last phase, ComQA maps the query graphs with KB. Finally, it evaluates the matchings to generate the answer. The ranking is based on semantic similarity.

ISOFT is a hybrid system, that is, it seeks an answer to unstructured and structured data. It divides the question into sub-queries. If the system does not find the answer to the data text or the answers are not appropriate (the system checks the answers using cosine similarity, Jaccard similarity) then it creates a SPARQL query for a triplet. When SPARQL query is used to find the answer, then lexical matching is used to match the predicate of the question with the KB properties. If the latter fails, semantic similarity applies.

Scalewelis system accepts controlled natural language, and therefore, this language has its syntax and semantics. The illustration of the question for the concrete syntax of LISQL2 (query language) is in the form of a tree (looks like a syntactic tree). The abstract syntax of LISQL2 represents the internal representation of the question. The illustration of the question for abstract syntax is in the form of a tree (the ontological elements of the question are displayed, e.g. class, property, and others). In order to create the sequential structure of the question, the focus must be defined.

Intui2 represents the question as a syntactic tree. Each node corresponds to a syntactic pattern. For each syntactic pattern, there is a mapping suggestion (it can be object or subject, RDF triplet, or complex SPARQL query) and the corresponding URIs are also specified. Ambiguities remain, and it is possible for the system to create many final questions if there are many interpretations of the question. At this point, we should mention that each synfragment corresponds to a subtree (the synfragment can be a concept URI, as an RDF triple or as a complex RDF query). The system assigns a degree to each synfragment. Complex synfragments are calculated by multiplying the ratings of their components. URI synfragments (Subject/object URIs and Predicate URIs) are graded based on string similarity.

FREyA when matching potential ontology concepts to KB data (called ontology concepts) may not be able to find the appropriate match (either not in the KB or the corresponding options are more than one). In this case, the system gives the user some options to choose from. System suggestions or recommendations are ranked. String similarity, based on the Monge Elkan metrics with Soundex algorithm, is used for the ranking, and Wordnet and Cyc were also used to find synonyms. Among the options is the option none in case no option fits. The system stores the user's choice to be used for system training to improve its performance over time.

SWIP system represents the question as a dependency tree and uses an intermediate language, called pivot language. The question is translated or converted to pivot query via query patterns. Finally, the question (which is translated to pivot query) corresponds to the elements of the knowledge base.

OAKIS tries to understand the user's question using triplets. The question is converted into relational patterns via a typed question. For each match (typed question with relational pattern), there is a similarity score.

SemSek represents the question as a dependency tree or graph. To match the terms of the sentence (the result from the question annotation) with data from DBpedia applies semantic matching using the semantic relatedness measure based on WordNet structure. A semantic similarity is relations of two concepts of the type: is-a.

RTV converts the question into a graph. The question items are matched with the KB items via probabilities. Unambiguous is resolved using a joint disambiguation approach and statistical inference.

WDAqua-core1 system accepts the question and converts it to a graph. Initially, all possible meanings (KB matches) for a word or meanings of the question can be retained in the graph. Then, the matches that are considered most likely are kept. The ambiguities persist until the creation of the SPARQL queries, which are classified.

IQA extracts the keywords from the question using a Shallow Parser. In the next step, the system predicts for each keyword of the previous step whether it is an entity or a relation. Then, for each keyword, a set of candidate matches are discovered. The matchings made between a question and a knowledge graph, that is, the matching of the entities and the relation of the question with the knowledge graph, create a graph in order to answer the user's question. The semantic matching of a question with the knowledge graph is based on an LSTM (Long Short-Term Memory) neural network. There are several SQA systems, which are based on neural networks to answer the question of a user. A thorough survey can be found in Chakraborty *et al*. (2021).

Intui3 is a neural network-based QAS over a knowledge graph. Intui3 uses SENNA for PoS tagging, chunking, and NER. SENNA is a deep neural network-based system. Intui3 uses the Stanford CoreNLP suite for lemmatization. The system combines the results from chunking and name entity recognition to combine smaller chunks into a larger chunk. Then, the system gives an interpretation or interpretations for each chunk. The interpretation of the user question is created by combining the interpretations of the chunks using rules, and for each interpretation, a score is given. The system selects the interpretation with the highest score, and the appropriate SPARQL query is created.

The system of Sorokin and Gurevych (Sorokin & Gurevych, 2017) is an end-to-end neural architecture to create the semantic representation of a user's question. The system accepts the question in natural language and converts it to a graph. It applies tokenization, POS tagging, and fragments extraction. Afterward, a list of Wikidata entities is found for each fragment, and then, the system creates possible semantic representations as graphs based on iterative representation generation. Essentially, it performs phrase mapping of the elements of the question with the elements of the KB. The candidate semantic graphs and the question are converted into fixed-size vectors based on a CNN model. The best graph is found based on cosine similarity.

The system of Athreya *et al*. (2021) is a neural network-based QAS over a knowledge graph using query templates. It accepts the question in natural language and performs classification into a corresponding template. Classification is achieved with the help of recursive neural networks. Firstly, dependency parsing and POS tagging are applied. Syntactic parsing is vectorized for the recursive neural network model. The Tree-LSTM is applied to learn the question. Also, the corresponding query templates are found. In particular, the top-n SPARQL templates are found. The candidate SPARQL templates are filled by mappings to the KB, representing semantically the user's question.

The system of Liang *et al*. (2021) is a Tree-LSTM neural network-based QAS over a knowledge graph. The system accepts the question of user in natural language and applies tokenization, POS tagging, lemmatization, and dependency parsing. The question type classification is performed via ML. Phrase mapping of the elements of the question to the elements of KB is applied. Candidate SPARQL queries are created and are ranked based on Tree-structured Long Short-Term Memory (Tree-LSTM). Finally, the system selects the most appropriate queries.

LAMA represents the semantic interpretation of the question in the form of a graph. Entity Extractor and Property Extractor work in parallel to match the question elements with the KB elements as some elements are based on the entity-property relations pattern. Parallel processing also decreases processing time.

According to the above, in SQA systems we observe that the dominant model of representation of the question is the semantic graph-based model which is justified by the fact that the stored linked data in KBs are in graph form. All SQA systems use semantic graph-based models, since the SPARQL questions also identify an abstract graph.

Examples of systems based on the probability model are gAnswer, RTV, CASIA@V2, Xser, QAnswer, WDAqua-core1and IQA. Examples of systems based on algebraic models are DENNA, Intui2, RTV, CASIA@V2, ISOFT, Xser, SemGraphQA, YodaQA, ComQA, LAMA, Sorokin and Gurevych (2017), Athreya *et al*. (2021) and Liang *et al*., (2021). There are systems that on top of the semantic graph-based model they use other models as well, such as probability or algebraic models. Examples of such systems are DENNA, gAnswer, Intui2, RTV, CASIA@V2, ISOFT, Xser, SemGraphQA, YodaQA, ComQA, WDAqua-core1, IQA, and LAMA. For example, Xser uses semantic graph-based, algebraic, and probability models. Finally, we could not find set theoretic models, as this model treats the documents as sets of words or phrases. However, we do not rule it out because it can be used in hybrid domain SQA systems.

3.6. Classification based on characteristics of the KB

Mishra and Jain (2016) report the following values: source size, language, heterogeneity, genre, and media, which are based on the characteristics of data sources. We replace the characteristics of data sources of Mishra & Jain with characteristics of the KB, and we identify as the values of this criterion the following: multilingualism (language in QASs) and distributed KB (heterogeneity in QASs). In SQA systems, we do not observe genre (genre characterizes the type/style, which can be formal or informal, of the language used) as the format of stored linked data cannot be formal or informal like the language of documents. Also, the answers like audio, video, and sound have not yet been integrated in the SQASs. Of course, this task is difficult even in QASs. SQA systems could also be sorted by source size as open domains are larger than closed domains, but we have selected the classification as open domain, closed domain, and hybrid domain. Therefore, we have replaced the criterion of the characteristics of data sources of Mishra and Jain (2016) with the criterion of the characteristics of the KB.

3.6.1 Multilingualism

Multilinguality has gained a lot of interest mainly for two reasons: a large number of actors are created and published as open data not only in English but also in other languages and on the other hand the number of users who want to access this data and do not have English as their mother tongue, and it is constantly increasing. DBpedia has multilingual labels, and versions of DBpedia such as the Spanish and the French DBpedia are available (Lopez *et al*., 2013).

Multilinguality is also considered as an evaluation criterion in QALD (Lopez *et al*., 2013; Cimiano *et al*., 2013). There are systems that only support one language, such as Alexandria, which only accepts questions in German. Most systems accept questions in English. The SWIP system accepts the user's question and converts it to an intermediate query called pivot query, that is, it is converted into an intermediate language called pivot language. This intermediate language helps with multilingualism. There is a separate section for each language that converts the user question to pivot language but the section referring to the formation of the pivot query remains the same in each case. However, at the same time it states that it accepts questions only in English. WDAqua-core1 accepts questions in 5 different languages (English, French, German, Italian, and Spanish). The LAMA is a multilingual system for questions in English or French. The semantic representation of the user question is based on a set of lexico-syntactic patterns for entity and property extraction. This pattern-based method helps the system to be multilingual as the patterns are in different languages. QAnswer (Diefenbach *et al*., 2019) supports multilingualism and accepts questions in English, German, French, Italian, Spanish, Portuguese, Arabic, and Chinese. The GFMed system is able to question in English and Romanian.

3.6.2 Distributed Knowledge Base

There are also SQA systems that look for the answer to the user's question in many KBs. PowerAqua accepts a question in natural language and returns ranked answers, which are based on many KSs. This system is responsible for detecting the appropriate KB that is relevant to the user question. Another system, which connects many linked data resources to create SPARQL queries and gives the answer to the user, is SMIQ. QAnswer (Diefenbach *et al*., 2019) can query at the same time three different KBs (such as Wikidata with Lexemes, LinkedGeodata, and Musicbrainz.) On the other hand, there are systems that can use more than one KB but not simultaneously like DEANNA, SemGraphQA, and WDAqua-core1. GFMed and POMELO are closed domain systems, which are querying biomedical linked data such as Diseasome, SIDER, and DrugBank. Both systems use the above three KBs to answer the question of users.

3.7. Classification based on interaction of user

We added the interaction of the user, and the values of this criterion are user interaction in disambiguation and type of user. In several systems (such as FREyA and IQA), we notice that the user does not only enter a question waits to view the answer from the system. But, instead the user plays an active role in the process of converting the natural language question into a structured query. In particular, in some systems, in order to disambiguate the user's question, they suggest options to the user to select the appropriate translation for some specific elements of the question. Another active role of the user is when there are systems that use controlled natural language. In such systems, the user is guided by the tips of the system in order to create the desired question. As we have mentioned, these systems were created to support ordinary users who are not familiar with SPARQL and face difficulties in searching linked datasets. But some of them serve both ordinary users and expert users, who can overview the conversion of the question and/or they can modify the resulting SPARQL query.

3.7.1. User interaction in disambiguation

One of the challenges that SQA systems face is ambiguity, that is, the same word has different interpretations (Höffner *et al*., 2017). There are systems that the user intervenes to carry out the disambiguation such as FREyA. FREyA may not find the right match for an element in question (either not in the KB or the options for the match are more than one); then, the system gives the user some options to choose from. System suggestions or recommendations are ranked. String similarity, based on the Monge Elkan metrics with Soundex algorithm, is used for the classification, and Wordnet and Cyc were also used to find synonyms. Among the options is the option, none in case no option fits. The system stores the user's choice to be used to train the system to improve its performance over time. In the disambiguation problem, the Swip system interacts with the user and selects the appropriate interpretation. The same happens with the IQA system, in order to disambiguate the correspondences or matching with elements of KB. The user interacts with the system in order to accept the appropriate interpretation of the question and consequently to create the appropriate SPARQL query.

Moreover, in the SQA systems that use controlled natural language there is an interaction between the system and the user in order to guide the user to create the user question such systems are Scalewesis, SQALl2SPARQL, and CANaLI. Essentially, this interaction indirectly intends to resolve the ambiguities since there are specific options for the user. Of course, the disambiguation is solved through controlled natural language, but the disadvantage of these systems is that the user must become familiar with controlled natural language.

3.7.2. Type of user

Also, another category is the type of the users. For example, there are systems aimed at casual users, which just enter questions and wait for the answer, and advanced users. Such systems are DEANNA,

which provides the ability to intervene in the translation process and observe how the components interact to create the final result. Another system that supports casual and expert users is Swipe. The Swipe system allows advanced users to view the query, modify the SPARQL query, and submit it. Of course, the current version of the system does not allow writing queries. Finally, the IQA system displays to the user the top-ranked SPARQL query. Furthermore, the system displays the interpretation of the user's question in natural language.

3.8. Classification based on answers

Mishra and Jain (2016) categorize QASs answers in extracted answer (such as sentences, paragraphs, multimedia) and generated answer (yes or no answers, dialogue answers, opinionated answers). In question answering system over linked data, the types of answers are boolean, date, numeral, resource, string. If we consider the Mishra & Jain criterion, then the generated answers are boolean and numeral. The numeral answer type can come from questions that look for a stored answer in the KB, for example, in questions like the following: 'How deep is Lake Placid?' This means that numeral is not generated, but rather extracted or retrieved. On the other hand, questions like 'How many programming languages are there?' in order to be answered, it is required to find all the programming languages in the KB and then count them. This means that the numeral is generated or calculated using a SPARQL aggregation function. On the other hand, the extracted answers are the date (if there is one registered in KB), resource, and string.

3.9. Techniques used in SQASs

Mishra and Jain (2016) identify one more category, called 'techniques used in QASs'. These techniques are related to response retrieval techniques such as data mining techniques searching for factual data, IR searching techniques for factual information in text documents, NLP techniques searching for information, knowledge retrieval searching for understanding, and creating knowledge. We replace the above techniques used in QASs with the techniques used in SQASs. Instead of data mining, that involves both some knowledge discovery algorithms and several data preparation techniques, we use the broader term ML that includes more algorithms to learn knowledge or patterns from both structured and unstructured information. These techniques in SQA systems are ML, IR, NLP, and knowledge retrieval and discovery.

3.10. Overview of system classification

The criteria of QASs that were developed by Mishra and Jain (2016) are application domain, types of questions, types of analyses of questions, types of data, characteristics of data sources, types of representations used for questions, and types of representations used for questions and their matching functions, types of techniques used for retrieving answers, and forms of answers generated. The classification we created is based on types of domains, types of data sources, types of questions, types of analysis done on questions, types of representations used for questions and types of representations used for questions and their matching functions, characteristics of the KB, types of techniques used for retrieving answers, user interaction, and answers. Summarizing, the differences in our classification criteria with those of Mishra and Jain (2016) can be grouped into 3 categories:

1. differentiated criteria (in terms of the values they get), which are indicated with an asterisk in Table 1,
2. criteria not included in Mishra and Jain (2016) but included in our survey because they are used in SQAs, which are indicated with italic and boldface font in Table 1, and

3. criteria included in Mishra and Jain (2016) that have been replaced in our survey with other, more appropriate, criteria, indicated with boldface in Table 1.

Differences and similarities were discussed and described in detail in each category. Table 2 summarizes all the systems and classifies them according to the respective criteria.

4. Semantic question answering systems

In this section, we have chosen to describe some SQA systems to show the different systems architecture and the originality of each system. In this section, we give a detailed description of some systems to understand the originality and different architecture of SQA systems so that researchers can see the pros and cons and suggest solutions for complex SPARQL queries. Researchers or readers can also reuse or customize techniques or tools presented in SQA systems. The systems are categorized based on the competitions that are held, that is, based on the evolution of the systems. Competitions (QALD) increase the demands and challenges of systems (they add additional criteria) so we are talking about systems' evolution. We describe systems until the competition QALD-5. However, after the QALD-5 competition, most of the systems have already participated in previous competitions. This is the reason that in Section 4, we have 5 Subsections 4.1–4.5. Some systems that appeared in a later-than-the-5th QALD competition and/or systems that have not participated in any QALD competition, but have a merit worth describing it, have been included in Section 4.6.

4.1. Systems evaluated in QALD-1

FREyA (Feedback, Refinement and Extended Vocabulary Aggregation) is a question answering system over linked data (Damljanovic *et al*., 2010). The specific domain of the system is the Mooney Geoquery. The system was also evaluated on DBpedia and MusicBrainz. FREyA accepts a question in natural language from the user and gives the answer by searching for it through the KB. To find the answer, try to match the terms or words of the question with elements (for example classes, instances, properties, literals) from the ontology. Before matching, the system applies syntactic parsing to the question. Then takes the result from the syntactic parsing and combines it with heuristic rules to find potential ontology concepts from the question to map with elements of KB. Stanford Parser is used for syntactic analysis. Once the potential ontology concepts are found, they are matched with the data from the KB (which are called ontology concepts). It is possible for a potential ontology concept the system cannot find the right match (either it does not exist in the KB, or the matches are more than one). In this case, the system gives the user some options to choose. System suggestions or recommendations are classified. String similarity, based on the Monge Elkan metrics with Soundex algorithm, is used for the ranking. Wordnet[12] and Cyc were also used to find synonyms. Among the options is the option 'none' in case no option fits. The system stores the user's choice to be used to train the system to improve its performance over time. After matching, the SPARQL query is generated. To find the answer, the answer type of the question is specified.

PowerAqua (Lopez *et al*., 2012) accepts a question in natural language and returns classified answers, which are based on many KBs. The system implements a pipeline framework, which consists of four steps. In the first step, called the Linguistic Component, the system performs a linguistic analysis to convert the question into triplets with interdependencies and then marks and then marks each term of the triplet with one of the following designations: subject, predicate, or object. Each triplet has the following format: <subject, predicate, object>. The system uses the Gate NL processing tool to create this representation. The second step is called Element Mapping Component (PowerMap). This step consists of two components. The first component is responsible for detecting the appropriate KBs that

[12] https://wordnet.princeton.edu.

Table 2. *Classification of SQA systems and frameworks.*

System/ Framework	Types of domains (Open, Closed, Hybrid)	Types of data sources (Structured, hybrid)	Types of questions (Factoid, list, boolean, count)	Types of analysis done on questions (Morphological, syntactical, dependency, semantic, expected answer type, focus recognition of questions, pragmatic and discourse)	Types of representations used for questions and their matching functions (Algebraic, probability, semantic graph-based, theoretic)	of the KB (multilingualism, distributed knowledge base)	Interaction of user(In disambiguation, type of users)	Answers(Date, resource, string, boolean, numeral)	Techniques for retrieving answers (ML, IR, NLP, knowledge retrieval and discovery)
FREyA	Hybrid	Structured	Factual, list	Syntactical, semantic, expected answer type	Semantic graph-based		User Interaction in disambiguation	String, numeral, resource	Knowledge retrieval and discovery, ML, NLP
PowerAqua	Open	Structured	Factual, list	Syntactical, dependency, semantic	Semantic graph-based	Distributed knowledge base		String, numeral, resource	Knowledge retrieval and discovery, IR, NLP
SWIP	Closed	Structured	Factual, list, boolean, count	Dependency, semantic, focus recognition of questions	Semantic graph-based		User Interaction in disambiguation	String, boolean, numeral, resource	Knowledge retrieval and discovery, IR, NLP

Table 2. *Continued.*

System/ Framework	Types of domains (Open, Closed, Hybrid)	Types of data sources (Structured, hybrid)	Types of questions (Factoid, list, boolean, count)	Types of analysis done on questions (Morphological, syntactical, dependency, semantic, expected answer type, focus recognition of questions, pragmatic and discourse)	Types of representations used for questions and their matching functions (Algebraic, probability, semantic graph-based, theoretic)	of the KB (multilingualism, distributed knowledge base)	Interaction of user(In disambiguation, type of users)	Answers(Date, resource, string, boolean, numeral)	Techniques for retrieving answers (ML, IR, NLP, knowledge retrieval and discovery)
SWIP	Closed	Structured	Factual, list, boolean, count	Dependency, semantic, focus recognition of questions	Semantic graph-based		User Interaction in disambiguation	String, boolean, numeral, resource	Knowledge retrieval and discovery, IR, NLP
Alexandria	Open	Structured	Factual, list, boolean, count	Morphological, syntactical, dependency, semantic, expected answer type	Semantic graph-based			String, boolean, numeral, resource	Knowledge retrieval and discovery, IR, NLP
OAKIS	Open	Structured	Factual, list, count	Morphological, semantic, expected answer type	Semantic graph-based			String, boolean, numeral, date, resource	Knowledge retrieval and discovery, NLP

Table 2. *Continued.*

System/ Framework	Types of domains (Open, Closed, Hybrid)	Types of data sources (Structured, hybrid)	Types of questions (Factoid, list, boolean, count)	Types of analysis done on questions (Morphological, syntactical, dependency, semantic, expected answer type, focus recognition of questions, pragmatic and discourse)	Types of representations used for questions and their matching functions (Algebraic, probability, semantic graph-based, theoretic)	of the KB (multilingualism, distributed knowledge base)	Interaction of user(In disambiguation, type of users)	Answers(Date, resource, string, boolean, numeral)	Techniques for retrieving answers (ML, IR, NLP, knowledge retrieval and discovery)
SemSek	Open	Structured	Factual	Syntactical, dependency, semantic	Semantic graph-based			String, numeral, resource	Knowledge retrieval and discovery, NLP
DENNA	Open	Structured	Factual	Dependency, semantic	Semantic graph-based, algebraic		Type of users	String, numeral	Knowledge retrieval and discovery, NLP
gAnswer	Open	Structured	Factual, list, boolean, count	Dependency, semantic	Semantic graph-based, probability			String, boolean, numeral, resource	Knowledge retrieval and discovery, information retrieval, ML, NLP

Table 2. *Continued.*

System/ Framework	Types of domains (Open, Closed, Hybrid)	Types of data sources (Structured, hybrid)	Types of questions (Factoid, list, boolean, count)	Types of analysis done on questions (Morphological, syntactical, dependency, semantic, expected answer type, focus recognition of questions, pragmatic and discourse)	Types of representations used for questions and their matching functions (Algebraic, probability, semantic graph-based, theoretic)	of the KB (multilingualism, distributed knowledge base)	Interaction of user(In disambiguation, type of users)	Answers(Date, resource, string, boolean, numeral)	Techniques for retrieving answers (ML, IR, NLP, knowledge retrieval and discovery)
Intui2	Open	Structured	Factual, count	Morphological, syntactical, semantic, expected answer type	Semantic graph-based, algebraic			String, numeral, date, resource	Knowledge retrieval and discovery, IR, NLP
RTV	Open	Structured	Factual, list, boolean, count	Syntactical, dependency, semantic	Semantic graph-based, algebraic, probability			String, boolean, numeral, date, resource	Knowledge retrieval and discovery, IR, NLP
Scale-welis	Open	Structured	Factual, boolean	Syntactical, dependency, semantic, focus recognition of questions	Semantic graph-based		User Interaction in disambiguation	String, numeral boolean, resource	Knowledge retrieval and discovery, NLP

Table 2. *Continued.*

System/ Framework	Types of domains (Open, Closed, Hybrid)	Types of data sources (Structured, hybrid)	Types of questions (Factoid, list, boolean, count)	Types of analysis done on questions (Morphological, syntactical, dependency, semantic, expected answer type, focus recognition of questions, pragmatic and discourse)	Types of representations used for questions and their matching functions (Algebraic, probability, semantic graph-based, theoretic)	of the KB (multilingualism, distributed knowledge base)	Interaction of user(In disambiguation, type of users)	Answers(Date, resource, string, boolean, numeral)	Techniques for retrieving answers (ML, IR, NLP, knowledge retrieval and discovery)
SQUALL2S-PARQL	Open	Structured	factual, list, booelan, count	Syntactical, semantic	Semantic graph-based		User Interaction in disambiguation	String, boolean, numeral, date, resource	Knowledge retrieval and discovery, NLP
CASIA-@V2	Open	Structured	Factual, list, booelan, count	Dependency, semantic	Semantic graph-based, algebraic, probability			String, boolean, numeral, resource	Knowledge retrieval and discovery, IR, NLP
ISOFT	Open	Hybrid	Factual, list, boolean	Dependency, semantic, expected answer type, focus recognition of questions	Semantic graph-based, algebraic			String, numeral	Knowledge retrieval and discovery, IR, NLP

Table 2. *Continued.*

System/ Framework	Types of domains (Open, Closed, Hybrid)	Types of data sources (Structured, hybrid)	Types of questions (Factoid, list, boolean, count)	Types of analysis done on questions (Morphological, syntactical, dependency, semantic, expected answer type, focus recognition of questions, pragmatic and discourse)	Types of representations used for questions and their matching functions (Algebraic, probability, semantic graph-based, theoretic)	of the KB (multilingualism, distributed knowledge base)	Interaction of user(In disambiguation, type of users)	Answers(Date, resource, string, boolean, numeral)	Techniques for retrieving answers (ML, IR, NLP, knowledge retrieval and discovery)
SMIQ	Open	Structured	Factual, list, booelan, count	Dependency, semantic	Semantic graph-based	Distributed knowledge base		String, boolean, numeral, date, resource	Knowledge retrieval and discovery, NLP
Swipe	Open	Structured	List, count		Semantic graph-based		Type of users	String, boolean, numeral, date, resource	Knowledge retrieval and discovery
Xser	Open	Structured	Factual	Dependency, semantic	Semantic graph-based, algebraic, probability			String, numeral	Knowledge retrieval and discovery, IR, ML, NLP

Table 2. *Continued.*

System/ Framework	Types of domains (Open, Closed, Hybrid)	Types of data sources (Structured, hybrid)	Types of questions (Factoid, list, boolean, count)	Types of analysis done on questions (Morphological, syntactical, dependency, semantic, expected answer type, focus recognition of questions, pragmatic and discourse)	Types of representations used for questions and their matching functions (Algebraic, probability, semantic graph-based, theoretic)	of the KB (multilingualism, distributed knowledge base)	Interaction of user(In disambiguation, type of users)	Answers(Date, resource, string, boolean, numeral)	Techniques for retrieving answers (ML, IR, NLP, knowledge retrieval and discovery)
CANaLI	Hybrid	Structured	Factual, list, booelan, count	Syntactical, semantic	Semantic graph-based		User Interaction in disambiguation	String, boolean, numeral, date, resource	Knowledge retrieval and discovery, NLP
QAnswer	Open	Structured	Factual, list, boolean, count	Morphological, dependency, semantic, expected answer type	Semantic graph-based, probability			String, boolean, numeral, date, resource	Knowledge retrieval and discovery, IR, NLP
SemGraphQA	Open	Structured	Factual, boolean, count	Syntactical, dependency, semantic, expected answer type	Semantic graph-based, algebraic			String, boolean, numeral, date, resource	Knowledge retrieval and discovery, IR, NLP

Table 2. *Continued.*

System/ Framework	Types of domains (Open, Closed, Hybrid)	Types of data sources (Structured, hybrid)	Types of questions (Factoid, list, boolean, count)	Types of analysis done on questions (Morphological, syntactical, dependency, semantic, expected answer type, focus recognition of questions, pragmatic and discourse)	Types of representations used for questions and their matching functions (Algebraic, probability, semantic graph-based, theoretic)	of the KB (multilingualism, distributed knowledge base)	Interaction of user(In disambiguation, type of users)	Answers(Date, resource, string, boolean, numeral)	Techniques for retrieving answers (ML, IR, NLP, knowledge retrieval and discovery)
YodaQA	Open	Hybrid	Factual	Dependency, semantic, expected answer type, focus recognition of questions	Semantic graph-based, algebraic			String, date, numeral	Knowledge retrieval and discovery, IR, ML, NLP
ComQA	Open	Structured	Factual, list, booelan, count	Dependency, syntactical, semantic	Semantic graph-based, algebraic			String, boolean, numeral, date, resource	Knowledge retrieval and discovery, IR, NLP
WDAquacore1	Hybrid	Structured	Factual, list	Morphological, semantic	Semantic graph-based, probability	Multilingualism		String, numeral date, resource	Knowledge retrieval and discovery, IR, NLP

Table 2. *Continued.*

System/ Framework	Types of domains (Open, Closed, Hybrid)	Types of data sources (Structured, hybrid)	Types of questions (Factoid, list, boolean, count)	Types of analysis done on questions (Morphological, syntactical, dependency, semantic, expected answer type, focus recognition of questions, pragmatic and discourse)	Types of representations used for questions and their matching functions (Algebraic, probability, semantic graph-based, theoretic)	of the KB (multilingualism, distributed knowledge base)	Interaction of user(In disambiguation, type of users)	Answers(Date, resource, string, boolean, numeral)	Techniques for retrieving answers (ML, IR, NLP, knowledge retrieval and discovery)
IQA	Open	Structured	Factual, list, boolean	Syntactical, semantic, expected answer type	Semantic graph-based, probability		User Interaction in disambiguation, type of users	String, numeral, boolean, resource	Knowledge retrieval and discovery, ML, IR, NLP
LAMA	Open	Structured	Factual, list, booelan, count	Syntactical, semantic, expected answer type	Semantic graph-based, algebraic	Multilingualism		Date, resource, string, boolean, numeral	Knowledge retrieval and discovery, IR, NLP
Intui3	Open	Structured	Factual, list, count	Morphological, syntactical, semantic	Semantic graph-based, algebraic			Resource, numeral, string	Knowledge retrieval and discovery, ML, NLP

Table 2. *Continued.*

System/ Framework	Types of domains (Open, Closed, Hybrid)	Types of data sources (Structured, hybrid)	Types of questions (Factoid, list, boolean, count)	Types of analysis done on questions (Morphological, syntactical, dependency, semantic, expected answer type, focus recognition of questions, pragmatic and discourse)	Types of representations used for questions and their matching functions (Algebraic, probability, semantic graph-based, theoretic)	of the KB (multilingualism, distributed knowledge base)	Interaction of user(In disambiguation, type of users)	Answers(Date, resource, string, boolean, numeral)	Techniques for retrieving answers (ML, IR, NLP, knowledge retrieval and discovery)
GFMed	Closed	Structured	Factual, list, booelan, count	Syntactical, semantic	Semantic graph-based	Distributed knowledge base, multilingualism		String, boolean, numeral	Knowledge retrieval and discovery, ML, NLP
POM-ELO	Closed	Structured	Factual, list, booelan, count	Morphological, syntactical, semantic, expected answer type, focus recognition of questions	Semantic graph-based	Distributed Knowledge Base		Resource, string, boolean, numeral	Knowledge retrieval and discovery, NLP
Sorokin and Gurevych (2017)	Open	Structured	Factual	Semantic	Algebraic, semantic graph-based			String, numeral	Knowledge retrieval and discovery, ML, IR, NLP

Table 2. *Continued.*

System/ Framework	Types of domains (Open, Closed, Hybrid)	Types of data sources (Structured, hybrid)	Types of questions (Factoid, list, boolean, count)	Types of analysis done on questions (Morphological, syntactical, dependency, semantic, expected answer type, focus recognition of questions, pragmatic and discourse)	Types of representations used for questions and their matching functions (Algebraic, probability, semantic graph-based, theoretic)	of the KB (multilingualism, distributed knowledge base)	Interaction of user(In disambiguation, type of users)	Answers(Date, resource, string, boolean, numeral)	Techniques for retrieving answers (ML, IR, NLP, knowledge retrieval and discovery)
Athreya *et al*. (2021)	Open	Structured	Factual, boolean, count	Syntactical, dependency, semantic, expected answer type	Algebraic, semantic graph-based			String, boolean, numeral	Knowledge retrieval and discovery, ML, IR, NLP
Liang *et al*. (2021)	Open	Structured	Factual, list, booelan, count	Morphological, syntactical, dependency, semantic, expected answer type	Algebraic, semantic graph-based, probability			Resource, string, boolean, numeral	Knowledge retrieval and discovery, ML, IR, NLP

are relevant to the user's question. In order to perform the matching of the query elements with the data from the KBs, the system applies syntactic techniques. Also, in this step the WordNet tool is used to find synonyms of the words in the question. However, it is possible for a term of the question to have more than one interpretation. This disambiguation problem is solved using Word Sense Disambiguation techniques. This is Semantic Validation Component which consists of the second component. The third step is called Triple Mapping Component (Triple Similarity Service—TSS). The purpose of the third step is to find the most appropriate interpretation for the user's question. The system tries to find out which ontology triplets best correspond to the triplets of the user question. The filtering heuristics are used to decrease the set of candidates. The last step is called Merging and Ranking Component. The sets of ontological triples are merged, and answers are composed; then, the results are ranked based on criteria.

The SWIP (Semantic Web Interface Using Patterns) system consists of two steps (Pradel *et al*., 2012, 2013). Swip is a semantic Web interface using patterns. In the first step, the system accepts the user question and converts it to an intermediate query called pivot query, that is, it is converted to an intermediate language called pivot language. This intermediate language helps with multilingualism. There is a separate section for each language that converts the user question to pivot but the section, which forms pivot query, remains the same in each case. The name entities are specified. The dependency parser extracts the dependency tree of the user question. The dependency parsing is taken into consideration name entities. The parser, that uses the system, is MaltParser. Then, it identifies what the user is asking for example if the user is asking to find a date or is looking for a film. Finally, query patterns are used to create the pivot query. The elements of the question which are translated into the pivot query correspond to elements of the KB. In the disambiguation problem, the system interacts with the user who selects the appropriate interpretation. Finally, the SPARQL query is created.

4.2. Systems evaluated in QALD-2

Alexandria is a German question answering system over linked data (Wendt *et al*., 2012). Essentially, the system consists of two distinct steps. An ontology was created for this system based on Freebase, Dbpedia, and user-generated content. The user question is represented as a dependent graph using the dependency parser, MaltParser. This perseris trained on the German Tiger corpus. Linguistic analysis also includes lemmatization namely morphological analysis. Name entities are identified using an index. In case of ambiguity, the user selects the appropriate option. Apart from identification of name entities, Alexandria detects dates and times using the open-source date parser provided by the Yago project to German. For other terms of the question are used hand-crafted lexica. Before the end of the first step, the system matches the terms of the linguistic analysis with terms from the ontology. In the second step, Alexandria gives the semantic description for the terms of the graph (e.g. for the term Angelina Jolie the semantic description is Person). After the process of the semantic description, the composition of semantic descriptions is implemented in order to create the SPARQL query.

OAKIS (Cabrio *et al*., 2012) tries to answer user questions based on relational patterns. Specifically, the WikiFramework repository was created by exporting relational patterns from Wikipedia and DBpedia. The relational patterns have the following form: S is subject (domain) of the relation, O is the object (range), and P (predicate or property) is the name of the relation. The system consists of two main components. The first component is called query generator. The purpose of this component is to create typed questions that correspond to relational patterns. It creates the SPARQL query from relational patterns. More specifically, this component is responsible for identifying name entities and the expected answer type. There are three options available for the name entity. If the first option fails, the system proceeds to the next option. If the system does not find a name entity with the second option, then it uses the last option. In the first option, the system uses the Named Entity Recognizer in Stanford Core NLP (if the question contains a phrase name entity that exists in DBpedia, and this phrase name

entity is in the results of Recognizer then the largest phrase name entity is retained). In the second option, Stanford looks for the name entity in proper noun; then in this case, it gets the longest label from DBpedia. In the latter case, it looks search in DBpedia to find the longest instance label. After the system identifies the named entity and the expected answer type then it creates the typed question. The second component is called matcher pattern. This accepts typed questions and corresponds to patterns. Specifically, a set of patterns are retrieved for each typed question. One SPARQL query or two queries are created for each pattern. If exists some SPARQL queries, then the query with the highest score is selected. The query must have at least at least one result.

SemSek (Aggarwal & Buitelaar, 2012) has a pipeline architecture and consists of three basic steps. The first step is linguistic analysis. The system accepts the user's question and performs dependency analysis using Stanford Parser. Also, in the linguistic analysis are identified the words that should not be separated but they are considered phrases (the result of this analysis is similar to the result of the name entity recognizer). The goal of the step is to create an ordered list as follows: the first term of the list is the center term of the dependent graph, and the next term is the term that depends on the first term and so on. The next step is called query annotation. The system identifies DBpedia instances and classes in the ordered list using Lucene index (one for instances and one for classes). The disambiguation problem is resolved by retrieving wikiPageRedirects URIs. The last step is called Semantic Similarity and Relatedness. SemSek matches the terms of the sentence (the result from the question annotation) with data from DBpedia using semantic similarity and relatedness.

4.3. Systems evaluated in QALD-3

DENNA (DEep Answers for MaNy Naturally Asked questions) system can search for information in many KBs such as Yago, Dbpedia, Freebase, or other Linked Data sources (Yahya *et al*., 2012). The system is used by both ordinary and advanced users. The system architecture consists of six steps. The first step is called Phrase detection. The goal of the first step is to find the semantic items (such as relations, entities, and classes) of the user's question using detectors. Each detector is responsible for discovering a specific category of semantic item. Detectors use different named entity recognition and different techniques to determine relations. All detectors operate independently, and this leads to duplication (overlap) and ambiguity problems. The second step is called phrase mapping. The basic function of the second step is to match the phrases (the result of the first step) with elements from the KB using dictionaries. It is possible to identify multiple candidate matches for a phrase. The third step is called Q-unit generation. In this step, the system synthesizes the phrases to create triplets (triploids) using the dependency parser. Triplets have a form similar to the following: subject property object (i.e. they consist of two arguments and a relation that connects them). Q-units are created from combination of triploids and candidate phrases. The fourth step is called joint disambiguation. This step is responsible for resolving the ambiguities. The solution is based on creating or exporting a subgraph from the weighted disambiguation graph using an integer linear program (ILP) and the Gurobi ILP solver. The fifth step is called Semantic item grouping. The system creates semantic triples based on q-units and constraints of different semantic items. The last step is called Query generation, and it builds SPARQL queries from semantic triples.

gAnswer (Zou *et al*., 2014) consists of two basic phases in order to give an answer to the user. The system follows a data-driven approach. The first phase is called question understanding and the second query evaluation. In the first phase, the goal of the system is to create a semantic query graph Q^s. Initially, the system accepts the user's question. A dependency tree is created by the question using Stanford Parser. Then, the relations that exist in the tree are extracted using a dictionary. The last step for the first phase is the connection of semantic relations and the creation of semantic query graph Q^s from them. In this step, solutions for coreference resolution are used. The query evaluation is responsible for finding a subgraph (in RDF graph G) that matches Q^s to find the user's answer. In the second phase, a first step is the mapping of Q^s (the user's question is represented as Q^s) with the elements of the KB. At this point, all ambiguities are maintained. This point is the most important part of the system as disambiguation

problem is determined in the query evaluation step. This implementation leads to improve precision, and performance is significantly speeded. It is possible to find many subgraphs for Q^s and each one has a score based on confidence probabilities. The system detects top-k subgraphs, and this is NP-hard problem. Each subgraph implies an answer to the question; at this point, the ambiguities have been resolved. A subgraph in RDF graph matches Q^s, although only it is isomorphic to Q^s, and this is the answer to the user's question.

Intui2 (Dima, 2013) system consists of four phases. The first phase is called the Preprocessing Phase. Initially, the system accepts the user's question. The syntactic tree is constructed by the question using the Stanford CoreNLP. In the linguistic analysis of the question, Intui2 applies tokenizing, lemmatization, and POS tagging. A tree has been created which has sub-trees. The information rank (IR) is also calculated for each token. A Wikipedia word frequency index was created to calculate the information rank. Each token is calculated according to this index. The rarest words have a high information rank. The second phase is called the Analysis Phase. The purpose of this phase is the recursive traversing of tree, starting from a specific node called the root node. To find the root node, the cumulative rank or CR (for each subtree) must be calculated from the information rank of the node leaves of the subtree. The root node is the one that has the highest CR then traversing the next nodes is based on the ascending order of the CRs. Each node corresponds to a syntactic pattern. For each syntactic pattern, there is a mapping suggestion (it can be object or subject, RDF triplet, or complex SPARQL query) and the corresponding URIs are also specified. Ambiguities remain; it is possible that the system creates many final questions if there are many interpretations of the question. At this point, we should mention that each synfragment corresponds to a subtree (the synfragment can be a concept URI, as an RDF triple or as a complex RDF query). The third phase is called Scoring Synfragments. The system assigns a score to each synfragment. Complex synfragments are calculated by multiplying the score of their components. URI synfragments (Subject/object URIs and Predicate URIs) are figured based on string similarity. The fourth phase is called Reranking Phase. The correctness for each query is calculated as follows: the number of correct answers based on the answer type/the total number of answers. Thus, the final score of each query is multiplied by the result of correctness. The final query is selected, and the answer is retrieved.

RTV (Roma Tor Vergata) is a question answer system over linked data based on the Hidden Markov Model (HMM) approach and more specifically on statistical inference in order to answer the user's question (Giannone *et al*., 2013). Initially, the system applies dependency parsing to the question and the dependency tree is constructed. Also, a DBpedia lexicon is available in the parser. The ontological elements of the tree are detected to create the Markov chain. This phase is called the HMM initialization stage. The second phase is called HMM modeling where the states (corresponds the KB elements with the ontological elements of the question), emissions (expresses the probabilities between the question elements and the KB elements), and transitions (expresses the probabilities between the states based on semantics) of the Markov chain are defined. The ambiguities at this point remain. The Lucene index is used at this stage. The next phase is called HMM decoding or Viterbi decoding. The aim of this phase is to find the best interpretation for the question and resolve the ambiguities using a joint disambiguation approach and statistical inference. The last phase is called SPARQL query compilation, and the system creates the SPARQL query and retrieves the answer.

Scalewelis (Guyonvarc *et al*., 2013) is based on Sewelis system. Sewelis cannot be used for large datasets such as Dbpedia and uses an intermediate query language called LISQL which is characterized as semi-natural and is inspired by SQUALL (controlled natural language). Scalewelis uses LISQL2 (new version of LISQL) to create the SPARQL query. LISQL2 determines the incremental construction of the query. The system does not accept the user question in natural language, but the question is constructed incremental by users, who select query elements to create the query. Fifteen production rules are used for the concrete syntax of LISQL2. The final question is constructed by making successive use of the production rules. The illustration of the question of the concrete syntax has the tree form (looks like a syntactic tree). The abstract syntax of LISQL2 represents the internal representation of the question. Each rule of concrete syntax of LISQL2 corresponds to a rule of abstract syntax. The

illustration of the question for abstract syntax has tree form (the ontological elements of the question are displayed, for example, class, property, and others). To create the SPARQL query, systems need a SPARQL semantics for LISQL2 to convert the abstract syntax into SPARQL query. In order to create the incremental structure of the question, the focus must be defined. The focus is the position from where the query elements will be integrated into the question. Initially, the system gives the user an initial query and asks the user to select the suggested query elements. There is a finite number for increments (classes, properties, inverse properties, variables, and nodes). In order to grow the question, that is, to construct the final question, the focus must be determined. Two elements are needed to determine the focus, that is, the focus operates like a pair of LISQL2 sub-query and the context of this sub-query. The sub-query context and query help to create the final query. The rules of abstract syntax are used to create the focus (sub-query and context of sub-query), which using semantics for LISQL2 is converted to LISQL2 query and in turn this is converted to SPARQL query. Results from focus are used to create increments. Scalewelis is not dependent on datasets because it is associated with SPARQL Endpoints.

SQUALL2SPARQL (Ferré, 2012, 2013a, 2013b) accepts a question in a controlled natural language which is formulated by the user. This is considered the disadvantage of the system, that is, the user needs to learn the SQUALL (Semantic Query and Update High-Level Language) language. The system accepts the question in SQUALL language, and it is converted into an intermediate logical representation and finally translated into a SPARQL query. The types of questions can be factual, list, boolean, and count question. Its vocabulary is created by URIs. This is a disadvantage as the questions in SQUALL language are not so close to natural language. On the other hand, the system does not perform lexical analysis, and for this reason, any KBs can be used. However, if a linguistic resource is available, it can be used using words instead of URIs. The syntactic and semantic analyses of SQUALL are implemented as a Montague grammar. The semantics of the question in SQUALL language is the intermediate translation, which is converted to SPARQL query.

4.4. Systems evaluated in QALD-4

CASIA@V2 (Shizhu *et al*., 2014) is a pipeline framework and consists of four phases. The first phase is called Phrase detection. The system in the first phase detects possible semantic item of the question of user. The system does not use NER in order not to lose important phrases and keeps all the n-grams as a candidate, in which it applies some rules. The final selection of the appropriate options for the semantic items from the question is made later. The second phase is called mapping phrase to semantic item. The second phase accepts the result of the first and the system corresponds to the semantic items with elements from KB. Various techniques and resources are used to match the semantic items in question with the KB semantic items. Entity matching is done using Wikipedia and not only DBpedia. For classes, the system uses the word2vec tool to calculate the similarity between the phrases of the question and the KB. PATTY and ReVerb are used to map properties. The system detects the relations of the semantic properties of DBpedia with relations patterns of resources (Patty and ReVerb), and then if the phrase of the question corresponds to a relation pattern, then the corresponding properties of DBpedia are the candidates. Ambiguities remain and are resolved at next stage. The third phase is called Feature extraction and joint inference. This phase is based on the MarkovLogic Network (MLN). The first step of the third phase is the Feature extraction. The system extracts the features from the question and the KB using the dependency tree, which was built by Standard Parser. The second step resolves the ambiguities of the previous steps. The results from the previous steps are formulated as first-order logic clauses in an MLN. These clauses are combined to resolve ambiguities in a unified way. The last phase is called SPARQL generation. A semantic item query graph is created from the inference results of the previous phase. After the semantic item, triples are joined and combined to create SPARQL query.

ISOFT (Park *et al*., 2015) is a question answer system over structured data (linked data) and unstructured data (multi-source tagged text database-text data). This system combines two approaches

knowledgebase-based question answering (KBQA) and information-retrieval-based question answering (IRQA) and is called hybrid. Initially, the system accepts the question and performs the question analysis. Question analysis is based on statistical and rule-based approaches. Specifically, tokenization, part of speech tagging, dependency parsing, keyword extraction, term extraction, and named entity (NE) extraction, which are ordinary NLP techniques, are executed by the system. These techniques are important not only for the semantic answer type (SAT) and for the selection of answer but also for the further processing of the question. ClearNLP4 is used for tokenization, PoS tagging, and dependency parsing by ISOFT. WordNet dictionary is used to find synonyms of verbs and nouns which are extracted during the execution of the term extraction. For NE extraction, Spotlight is applied which is responsible to map entities in question to entities of DBpedia. Also, Q2S analysis, lexical answer type (LAT) extraction, and phrase extraction are called oriented techniques and are used for the analysis of the question. Q2S analysis is rule-based approach and converts the interrogative or imperative sentence (question) to declarative. LAT restricts the type of response and helps the SAT, which is used to reduce the number of candidate answers. The system creates Apache Lucene queries in order to find answers to multi-information tagged text databases. It is possible for a user question to create more than one query which means that the question must be split. The answers from the sequential queries are combined to find the final answer. More specifically, the two rightmost phrases are combined for the first query. To find the answer from a query, the answer from the first query and the next rightmost phrase is used and so on. This process is repeated until the answer to the question is found. If the system does not find the answer in the data text or the answers are not appropriate (the system checks the answers using cosine similarity, Jaccard similarity), it creates a SPARQL query for a triplet. When SPARQL query is used to find the answer, then lexical matching is used to match the predicate of the question with the KB properties. If the latter fails, semantic similarity is applied.

SMIQ (Semantic Model for Interpreting English Queries) is based on a linked data-driven approach (Tran & Nguyen, 2016). It uses different KBs to find the correct answer to the user's question. Initially, the system accepts the user's question and applies a pre-processing to the question. Specifically, it detects the proper nouns or name entities so that during the parsing, which will be done in the next stage, they will not be lost. The SMIQ then performs dependency analysis considering the proper nouns. The result of the dependency analysis is a set of dependencies between the elements of the question. Inference rules are applied to the set of dependencies to create the semantic interpretation of the question, which is called SMIQ. The inference rules are manually determined on the set of training questions of the Task 1 of QALD-4. Prolog language was used to extract the meanings of the question from the inference rules and the set of dependencies. This structure (SMIQ), that has the form of Triple SPO <subject, predicate, object>, allows the user question to be linked to the KB components.

Swipe (Atzori & Zaniolo, 2012) is a middleware system and uses a different approach, called Search by Example. The user does not enter a question in natural language but uses Wikipedia's pages to enter query conditions in the Infobox of Wikipedia fields, and they are converted to SPARQL query. First, the user uploads a Wikipedia page, which has content similar to what the user is looking for. This example page looks like the original Wikipedia page. Infobox fields are activated for the user to enter values in the fields. The user then presses the appropriate button to send the query. These functions are implemented by the User Interface module (UI). The query is received by the Query Manager module (QM), which is responsible for converting the SPARQL query and returning Wikipedia pages that satisfy the user question.

Xser (Xu *et al*., 2014) is a pipeline framework to convert a natural language question to a SPARQL query and can be applied to many KBs. Initially, the system accepts the user's question and tries to detect possible semantic items of the question. A phrase detector was built to detect semantic items using structured perceptron. Then, a semantic parser was created, a transition-based DAG parsing algorithm, to extract relationships between semantic items more specifically to discover triplets (subject—predicate—object) considering the result of the previous step. The above constitute the first phase of the system. The goal of the first phase is to represent the user's question intention. Also, the system in the first phase is

independent of KB. The system in the second phase corresponds to the question elements (result of the previous phase) with the KB elements based on a structured perceptron model to resolve ambiguities. The feature of parser is that it uses a stack, a queue, which consists of the extracted phrases from the previous step and a series of actions. The goal of the parser is to create triplets, that is, to give a label (subject, object, predicate) for each phrase and to associate it with another label. The label is permanently defined when the queue is empty. A phrase can change different labels during the parsing process and finally to prevail the highest score for the label. The mapping for subject and object lies in probabilities using the Freebase search API while for predicate it is based on the Naive Bayes model for calculating probabilities. In the second phase, the system depends on KB.

GFMed (Marginean, 2017) is a multilingual system. The user can type questions in Romanian or English. Also, it is a controlled natural language system that queries biomedical linked data, such as Diseasome, SIDER, and DrugBank. The system accepts the question of a user, and it converts in SPARQL query using GF grammars. The GFMed system builds on application manual grammars with a GF. Additionally, the GF is a special purpose programming language and helps with multilingualism. GF grammars are distinguished in abstract and concrete grammars. Consequently, GFMed has an abstract syntax for the closed domain (Diseasome, SIDER, and DrugBank) and two concrete syntaxes, one for English and Romanian, and one for SPARQL. GFMed has three lexicons, two lexicons for English and Romanian and a lexicon for SPARQL. The user's question is converted in the abstract syntax and then in concrete syntaxes for creating a SPARQL query. GFMed can handle complex questions.

Intui3 (Dima, 2014) accepts a natural language query that must be syntactically correct. The system tries to create the interpretation of the question using semantic and syntactic information. To interpret the question, the system applies Frege's Principle of Compositionality, which means that the interpretation of a complex expression is created by the interpretations of the constituent expression which are combined through rules. The system performs tokenization, POS tagging, name entity recognition, and chunking. Intui3 uses SENNA (v3.0) for PoS tagging, chunking, and NER. SENNA is a deep neural network-based system. We refer readers to Chakraborty *et al*., (2021) for a thorough overview of neural network-based question answering systems over knowledge graphs. Intui3 uses the Stanford CoreNLP suite for lemmatization. The system combines the results from chunking and name entity recognition to combine smaller chunks into a larger chunk. This is the pre-processing step. The system then gives an interpretation or interpretations for each chunk. The interpretation of a chunk is based on the chunk's type, the semantic and syntactic information of the chunk. The interpretation of the user question is created by combining the interpretations of the chunk using rules, and for each interpretation, a score is given. The system selects the interpretation with the highest score, and the appropriate SPARQL query is created. The appropriate interpretation may not be created, and so the user's question will not be answered.

POMELO (Hamon *et al*., 2014) is a closed domain SQA system. It accepts a user's question about biomedical linked data, such as Diseasome, SIDER, and DrugBank. Also, POMELO is considered a distributed KB question answering system. The system implements a pipeline framework, which consists of four steps. POMELO is based on frames, namely RDF triples (subject, object, and predicates) are mapped to frame elements and frame predicates. In the first step, called pre-processing step, it performs linguistic and semantic annotations. Specifically, the system applies word segmentation, part of speech tagging, and lemmatization of the words with TreeTagger. In this step, it identifies semantic entities with TermTagger Perl module. Additionally, it uses the term extractor YaTeA to improve the results. In this step, syntactic analysis is applied. The second step is called question abstraction. In this step, POMELO considers basic information such as question topic, predicate and argument, identification, and result from definition (negation, aggregation, boolean query). This basic information helps to create the query structure. The third step is called query construction. POMELO accepts the result of the previous step and creates a SPARQL graph pattern. Finally, it generates a SPARQL query considering the query construction.

4.5. Systems evaluated in QALD-5

CANaLI (Context-Aware controlled Natural Language Interface) (Atzori *et al*., 2016; Mazzeo & Zaniolo, 2016a, 2016b) uses a controlled natural language and an autocompleter, which helps the user to type the question according to KB. These features help to resolve the ambiguity. The system operates as a finite state, which accepts tokens and has 12 states. For example, the user enters a phrase immediately the system sends a query to the Lucene index which sends acceptable tokens. According to the operation of the finite automata, these tokens must be semantically correct according to KB, have a type according to the current and previous state (i.e. be syntactically correct), and must be a phrase that matches the string. The returned answers or suggestions are displayed to the user, who must select one of those suggested by the autocompleter. If the user does not select one of the suggested options, then the system prevents the user from continuing. The system instructs the user how to complete the question. The system was used in the following datasets: DBpedia, MusicBrain, and the three biomedical KBs: DrugBank, Diseasome, and SIDER.

QAnswer (Ruseti *et al*., 2015) is a pipeline framework, and the user question is represented as a graph (in the last step a graph is selected). Wikipedia plays an important role in converting the question into a graph. First, the user enters the question. QAnswer accepts the question and uses Stanford CoreNLP, which generates the dependency tree. Then, the system needs to detect DBpedia resources in the text and the links between them. There are three categories of resources that the system detects: individuals, types, and properties. Also, different methods are applied and developed to detect each resource. The expressions for individuals that exist in DBpedia, in the form of Wikipedia, redirect integrated into a trie data structure. The system in the step of detecting individuals looks for this structure. The system selects the longest sequence of words for individual. In addition, the sorting of individuals is based on edit distance and importance. The importance shows the number of Wikipedia pages that are linked to the individual's page. At this point, the ambiguities are maintained. The types (usually in the form <individual> is a <type> and are in the first sentences of Wikipedia articles) are discovered by the Wikipedia article and integrated into another trie data structure (operates as an index to detect the types of text). The integrated expressions for the types are in the form of a triplet (vertices and an edge). The step of individual detection creates graphs, so the type-detection is executed for each graph. Another index was created for properties using Wikipedia. Properties' detection is based on the properties index. After detection steps, a number of graphs have been created. Each graph should be matched to the KB data. The system selects the graph with the highest score and creates the appropriate SPARQL query.

SemGraphQA (Beaumont *et al*., 2015) converts a question of user to a graph (or graphs) and generates SPARQL query. Initially, the system accepts the user question and detects the entity identification, type identification, and relation identification. For the entity identification, DBpedia Spotlight is used. The type labels are used for type identification. A dictionary was created which contains the variants of relations of DBpedia with the help of WordNet for relation identification process. Each detection has a score, and ambiguities are maintained. The syntactic dependency graph is generated by the Stanford parser. Syntactic dependency graph is not annotated with the semantic nature of each word (entity or relation). Then for each word of the dependency graph, its semantic nature (entity or relation) is determined. The ambiguities remain (many graphs are created). Each graph corresponds to the KB elements. The ambiguities, created by matching, remain. Graphs are sorted by score and SPARQL queries are created.

YodaQA (Baudiš, 2015, September) is a hybrid question answering system. This means that the answer is determined by structured and unstructured data (English Wikipedia, enwiki). It still uses two KBs: DBpedia and Freebase. The system follows a pipeline framework and is influenced by DeepQA. YodaQA consists of four phases. The first phase is called question analysis. The user question is applied pos-tagging, dependency parsing, and name entity recognition. The purpose of the question analysis is to find some features such as clues (keywords), focus (what the question asks for), and LAT (Lexical Answer Type). The result of the question analysis is used by the hybrid databases to generate candidate

answers. The second phase is called Answer Analysis and is responsible for discovering some features for each answer. The third stage is the merging of the answers, and the last phase is the scoring of the answers.

4.6. Other semantic question answering systems

ComQA (Jin *et al*., 2019) converts the user question into a graph (or graphs). The system consists of three phases: question parsing, query graph construction, and candidate subgraph evaluation. In the first phase, it accepts the user question and extracts entities with the use of DBpedia Spotlight. The system detects the relations with the use of Stanford Parser that creates the dependency tree. In this phase, a set of extracted entities is created and a set of extracted relations, considering the KB. The system also extracts RDF triples from the syntactic structure and the hidden structure of the dependency tree. In the second phase, the system creates query graphs by joining the RDF triples. In the last phase, ComQA corresponds to the query graphs with KB. Finally, it evaluates the matchings to generate the answer. The ranking is based on semantic similarity.

WDAqua-core1 (Diefenbach *et al*., 2018b, 2020) is a multilingual system in five different languages namely English, German, French, Italian, and Spanish. Also, it can query several KBs such as Wikidata, DBpedia, MusicBrainz, DBLP, and Freebase at the same time. The system consists of 4 steps: question expansion, query construction, query ranking, and response decision. In the first step, the system detects all entities, properties, and classes, which exist in the user's question based on some rules. It is possible for an element of the user question to have many possible meanings (KB mappings). In the second step, the system creates many SPARQL queries because there can be many interpretations of the question. In the third step, the system sorts the SPARQL queries. In the last step, it adds an additional confidence score to the first ranked query. If this calculation is smaller than threshold, then the whole candidate list does not fit the user's question, and therefore, the system will not answer the user question.

IQA (Interactive Query Construction) (Zafar *et al*., 2020.) is a system based on user interaction in order to create the appropriate interpretation to the user question and, consequently, the appropriate SPARQL query. IQA can support complex questions using a user interaction scheme. The system consists of four components: Shallow Parser, Entity Linker, Relation Linker, Query Builder. Initially, the Shallow Parser discovers keywords phrases from the user question. It essentially identifies entities and relations. The Entity Linker and the Relation Linker match results of Shallow Parser with elements of KB. It is possible for a keyword phrase of the user question to find more than one matches with elements of KB. The IQA system uses an LSTM neural network for semantic matching of the question with the knowledge graph. We refer readers to Chakraborty *et al*. (2021) for a thorough survey of neural network-based question answering systems over knowledge graphs. The Query Builder component accepts results from the Entity Linker and the Relation Linker and creates SPARQL queries, which are sorted. The system displays to the user the top-ranked SPARQL query. But also, the system displays the interpretation of the user's question in natural language. If the user agrees with the interpretation of the question that appeared, then the user presses acceptance. Otherwise, presses reject, and the system explains the reason for rejecting the specific interpretation. At the same time, on the left-hand side of the user interface, the system displays to the user the current interaction option. This interaction is expressed with a question (inquiry) and displays the answer to the inquiry. The user selects one of the following options: yes, no, and I do not know. The interaction option and the top-ranked query depend on user's feedback. The interaction continues until the user accepts the final query.

The LAMA (Language Adaptive Method for question Answering) (Radoev *et al*., 2018) is a multilingual system for questions in English or French. The semantic representation of the user query is based on a set of lexico-syntactic patterns for entity and property extraction. This pattern-based method helps the system to be multilingual as the patterns are in different languages. The components of the system are the following: a Question Type Classifier, an Entity Extractor, a Property Extractor, and a SPARQL Query Generator. Initially, the Question Type Classifier identifies the question type based on a keyword list. The Entity Extractor discovers possible candidate entities by matching the question words with the

KB elements. The system uses various criteria to find the most correct match. Also, it uses criteria for disambiguation of the matching. The Property Extractor works in parallel with the Entity Extractor. The first uses property lexicon and Word2Vec similarity vector to extract and match the question elements with the KB elements. Finally, the SPARQL Query Generator creates queries from the semantic representation of the question. This system supports complex questions (multiple entities and relations), analyzing each complex question into smaller queries. Although the SPARQL Query Generator creates many queries, the system is still able to reduce the number of unnecessary queries. It rejects queries that return null answers.

The system of Sorokin and Gurevych (Sorokin & Gurevych, 2017) is an end-to-end neural architecture to create the semantic representation of a user's question. Firstly, the system accepts the question in natural language. This SQA system uses Stanford CoreNLP toolkit for tokenization POS tagging. After that, it extracts fragments using rules. A list of Wikidata entities is found for each fragment. In the next phase, the system creates possible semantic representations as graphs based on iterative representation generation. Specifically, for each entity that exists in the user question the possible relations and constraints are discovered according to the KB. Essentially, in this step the elements of the question are matched with the elements of the KB. It is possible to create many graphs for an entity, and consequently, many semantic representations are created. In the next step, the system chooses the best match or interpretation based on a neural network model. The candidate semantic graphs and the question are converted into fixed-size vectors, based on a CNN model. The best graph is found based on cosine similarity. Finally, it generates the SPARQL query.

The system of Athreya *et al*. (2021) is a neural network-based QAS over a knowledge graph using query templates. It accepts the question of a user in natural language and performs classification into the corresponding template. Classification is achieved with the help of recursive neural networks. In the first phase, which is called question analysis, dependency parsing and POS tagging are applied using the English Stanford Neural Network dependency parser and POS-tagger. In the second phase, called Input Preparation, the result of the previous phase, syntactic parsing, is vectorized in order to be used by the recursive neural network, during the third phase of system. In this phase, the Tree-LSTM is applied to learn the question. Also, the corresponding query templates are found. In particular, the top-n SPARQL templates are found. The Slot Filling is the next phase. In Slot Filling, each candidate SPARQL template has three kinds of slots: resources, predicates, and ontology classes. These slots are filled by mapping to the KB and to represent semantically the user's question. The SPARQL queries are generated when slots are filled.

The system of Liang *et al*. (2021) is a Tree-LSTM-based neural network QAS over a knowledge graph. The system consists of five components. The first component is called question analysis. In the first component, the system accepts the question of the user in natural language and applies tokenization, POS tagging, lemmatization, and dependency parsing. Question type classification is the second component of system. In this component, it detects the type of the question using ML. Another task of this component is mapping the elements of the question to the elements of the KB. For the identification of resources, properties, and classes, the system user several phrase mapping systems such as DBpedia Spotlight, TagMe, EARL, Falcon, and RNLIWOD. The next component is Query generation. In Query generation, the 'WHERE' clause is determined and candidate SPARQL queries are generated. The Query ranking is the last component. This component accepts the list of candidate SPARQL queries and is responsible to rank these queries. The ranking is based on Tree-structured Long Short-Term Memory (Tree-LSTM). The system selects the most appropriate queries. Finally, it executes the generated query.

5. Discussion and conclusions

In the semantic Web, experienced users can use the SPARQL language and therefore search for answers in KBs. On the other hand, ordinary users, who do not know the SPARQL language or the KB structure,

face difficulties in searching linked datasets. SQA systems are solution to this problem. Many prototype SQA systems have been developed for different datasets. In SQA systems, users ask questions in natural language using their own terminology and receive a response generated by searching in an RDF KB.

So far, there is a survey in Höffner *et al*. (2017) that describes challenges and solutions for SQA systems. Also, a survey by Diefenbach *et al*. (2018a) referred to techniques and steps used in SQA systems. This article distinguishes categories of SQA systems based on criteria in order to lay the groundwork for a collection of common practices as no categories of SQA systems have been identified. This categorization can also serve as an archive of frameworks and systems where each system is classified according to the techniques that it uses for various criteria, such as such as types of questions, types of analysis done on questions, types of representations used for questions, and their matching functions. This can help developers or anyone interested to find out directly the technique or steps used by each system, or to benchmark her own system against existing ones.

The criteria that we created for SQA systems overlap. For example, the Casia@V2 system, as shown in Table 2, is classified for all criteria except for 'Characteristics of the KB and Interaction of user'. All systems must belong to one of the sub-categories of 'types of domains'. The same goes for the types of data sources criterion. Most of the SQA systems use structured data as they were created for this purpose, to look for answers in structured data. However, there are also few SQA systems that use both structured and unstructured data, which are called hybrid. We consider that when the question is converted from free text to SPARQL then the question also becomes graph-based since the SPARQL questions also identify an abstract graph. Therefore, SQA system uses semantic graph-based models. As for the other models such as algebraic, probability models, feature-based, and theoretic models (in hybrid SQASs), it depends on the architecture of the system whether it will use these criteria.

In terms of the types of questions and answers criteria, it depends on the system architecture. There are systems, such as SMIQ, that accept all types of questions (factoid, list, boolean, count question) and give all kinds of answers (date, resource, string, boolean, numeral). While others systems, like SemSek, may not be able to process all types of questions or may not be able to provide all kinds of answers. The 'answers' criterion applies to all SQA systems, but it also depends on the nature of the system, whether it can create all kinds of answers or answer all kinds of questions.

The same applies to the criteria Characteristics of the KB and Interaction of user. It depends on the system architecture whether it will use these criteria. All systems use semantic analysis. Many of the systems apply dependency analysis. Most systems use this type of analysis, as structured data are linked data. This does not mean that they do not apply other types of analysis, again depending on the system architecture. Many systems can apply many types of analysis.

We notice that in many SQA systems, the division of the user question into multiple KB queries is applied in order to answer the difficult or complex questions of the users. ComQA uses the above solution approach, that is, it divides the question into smaller queries and, finally, tries to combine RDF triples to answer the user's question. LAMA analyzes each complex question into smaller queries. ISOFT uses structured and unstructured data to solve complex questions, too; it usually divides questions (when they are large) into sub-queries. Concerning complex questions, each system uses its own approach, such as CANaLI that can handle complex questions as finite automata, while Swipe supports complex queries based on Search by Example.

The criterion of the 'characteristics' of the KB (multilingualism, distributed KB) is mainly met in more recent systems. Older systems also tackle multilingualism occasionally. For example, the system SWIP states that its architecture is suitable for multilingualism and states that it has tackled the task 1 of QALD-3, namely multilingual question answering. However, at the same time it states that it accepts questions only in English. What we have noticed from recent systems (existing also, of course, in some older ones) is that they focus more on the challenge of multilingualism such as LAMA, WDAqua-core1, and QAnswer (Diefenbach *et al*., 2019). We can observe a common element between LAMA and SWIP that these systems use patterns.

The 'interaction of user' criterion (user interaction in disambiguation, type of users) is also not always met, depending on the nature of the system, that is, there are systems that allow the user to engage in

disambiguation while other systems face this obstacle differently. IQA, a recent system, and FREyA, an older one, both use user interaction in disambiguation. Finally, most systems are created for casual users. IQA uses a user interaction scheme, that is, the user intervenes in order to disambiguate the question so that the system can answer user's question successfully. So far, we have seen that in SQA systems, user interaction is utilized for disambiguation. In the IQA system, user interaction is utilized to answer complex questions, having, of course, the same goal, namely to disambiguate words.

To summarize, we classified the SQASs according to the following criteria: types of domains, types of data sources, types of questions, types of analysis done on questions, types of representations used for questions and their matching functions, characteristics of the KB, types of techniques used for retrieving answers, interaction of user, and answers. This paper was inspired by these criteria by Mishra and Jain (2016). In conclusion, the differences in our classification criteria with those of Mishra and Jain (2016) can be grouped into 3 categories:

1. differentiated criteria (in terms of the values they get),
2. criteria not included in Mishra and Jain (2016) but included in our survey because they are used in SQAs, and
3. criteria included in Mishra and Jain (2016) that have been replaced in our survey with other, more appropriate, criteria.

Conflicts of interests. The author(s) declare none.

References

Aggarwal, N. & Buitelaar, P. 2012. A system description of natural language query over dbpedia. In *Proceedings of Interacting with Linked Data (ILD 2012), workshop co-located with the 9th Extended Semantic Web Conference,* Unger, C., Cimiano, P., Lopez, V., Motta, E., Buitelaar, P. & Cyganiak, R. (eds). May 28, 2012, Heraklion, Greece, 97–100.

Allam, A. M. N. & Haggag, M. H. 2012. The question answering systems: a survey. *International Journal of Research and Reviews in Information Sciences (IJRRIS)* **2**(3), 211–221.

Athreya, R. G., Bansal, S. K., Ngomo, A. C. N. & Usbeck, R. 2021. Template-based question answering using recursive neural networks. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC),* 195–198. IEEE.

Atzori, M., Gao, S., Mazzeo, G. M. & Zaniolo, C. 2016. Answering end-user questions, queries and searches on Wikipedia and its history. *IEEE Data Engineering* **39**(3), 85–96.

Atzori, M. & Zaniolo, C. 2012. Swipe: searching Wikipedia by example. In *Proceedings of the 21st International Conference on World Wide Web,* 309–312.

Baudiš, P. 2015. YodaQA: a modular question answering system pipeline. In POSTER 2015-19th *International Student Conference on Electrical Engineering,* 1156–1165.

Beaumont, R., Grau, B. & Ligozat, A. L. 2015. SemGraphQA@ QALD5: LIMSI participation at QALD5@ CLEF. In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation Forum,* 1–10.

Bouziane, A., Bouchiha, D., Doumi, N. & Malki, M. 2015. Question answering systems: survey and trends. *Procedia Computer Science* **73**, 366–375.

Cabrio, E. Palmero Aprosio, A., Cojan, J., Magnini, B., Gandon, F. & Lavelli, A. 2012. QAKiS at QALD-2. In *Proceedings of Interacting with Linked Data (ILD) 2012, Workshop at ESWC 2012, Heraklion, Greece,* 2012.

Cimiano, P., Lopez, V., Unger, C., Cabrio, E., Ngomo, A. C. N. & Walter, S. 2013. Multilingual question answering over linked data (qald-3): lab overview. In *International Conference of the Cross-Language Evaluation Forum for European Languages,* 321–332. Springer.

Chakraborty, N., Lukovnikov, D., Maheshwari, G., Trivedi, P., Lehmann, J. & Fischer, A. 2021. Introduction to neural network-based question answering over knowledge graphs. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **11**(3), e1389.

Damljanovic, D., Agatonovic, M. & Cunningham, H. 2010. Natural language interfaces to ontologies: combining syntactic analysis and ontology-based lookup through the user interaction. In *Extended Semantic Web Conference,* 106–120. Springer.

Diefenbach, D., Both, A., Singh, K. & Maret, P. (2020). Towards a question answering system over the semantic web. *Semantic Web* **11**(3), 421–439.

Diefenbach, D., Lopez, V., Singh, K. & Maret, P. 2018a. Core techniques of question answering systems over knowledge bases: a survey. *Knowledge and Information Systems* **55**(3), 529–569.

Diefenbach, D., Migliatti, P. H., Qawasmeh, O., Lully, V., Singh, K. & Maret, P. 2019. QAnswer: a question answering prototype bridging the gap between a considerable part of the LOD cloud and end-users. In *The World Wide Web Conference,* 3507–3510.

Diefenbach, D., Singh, K. & Maret, P. 2018b. Wdaqua-core1: a question answering service for RDF knowledge bases. In *Companion Proceedings of The Web Conference* 2018, 1087–1091.

Dima, C. 2013. Intui2: a prototype system for question answering over linked data. In *Working Notes for CLEF 2013 Conference.*
Dima, C. 2014. Answering natural language questions with Intui3. In *Working Notes for CLEF,* 1201–1211.
Dimitrakis, E., Sgontzos, K. & Tzitzikas, Y. 2020. A survey on question answering systems over linked data and documents. *Journal of Intelligent Information Systems* **55**(2), 233–259.
Dubey, M., Banerjee, D., Chaudhuri, D. & Lehmann, J. 2018. EARL: joint entity and relation linking for question answering over knowledge graphs. In *International Semantic Web Conference,* 108–126. Springer.
Dwivedi, S. K. & Singh, V. 2013. Research and reviews in question answering system. *Procedia Technology* **10**, 417–424.
Fader, A. 2014. *Open Question Answering* (Doctoral dissertation). University of Washington, Computer Science and Engineering.
Ferré, S. 2012. Squall: a controlled natural language for querying and updating RDF graphs. In *International Workshop on Controlled Natural Language,* 11–25. Springer.
Ferré, S. 2013a. SQUALL: a controlled natural language as expressive as SPARQL 1.1. In *International Conference on Application of Natural Language to Information Systems,* 114–125. Springer.
Ferré, S. 2013b. SQUALL2SPARQL: a translator from controlled english to full SPARQL 1.1. In *Working Notes for CLEF 2013 Conference*.
Giannone, C., Bellomaria, V. & Basili, R. 2013. A HMM-based approach to question answering against linked data. In *Working Notes for CLEF 2013 Conference.*
Guyonvarch, J., Ferre, S. & Ducassé, M. 2013. Scalable query-based faceted search on top of SPARQL endpoints for guided and expressive semantic search. In *Proceedings of the Question Answering over Linked Data Lab (QALD-3) at CLEF,* Valencia, Spain.
Hamon, T., Grabar, N., Mougin, F. & Thiessard, F. 2014. Description of the POMELO system for the Task 2 of QALD-2014. *CLEF (Working Notes)* **1212**, 28.
Höffner, K., Walter, S., Marx, E., Usbeck, R., Lehmann, J. & Ngonga Ngomo, A. C. 2017. Survey on challenges of question answering in the semantic web. *Semantic Web* **8**(6), 895–920.
Jin, H., Luo, Y., Gao, C., Tang, X. & Yuan, P. 2019. ComQA: question answering over knowledge base via semantic matching. *IEEE Access* **7**, 75235–75246.
Kolomiyets, O. & Moens, M. F. 2011. A survey on question answering technology from an information retrieval perspective. *Information Sciences* **181**(24), 5412–5434.
Liang, S., Stockinger, K., de Farias, T. M., Anisimova, M. & Gil, M. 2021. Querying knowledge graphs in natural language. *Journal of Big Data* **8**(1), 1–23.
Lopez, V., Fernández, M., Motta, E. & Stieler, N. 2012. Poweraqua: supporting users in querying and exploring the semantic web. *Semantic Web* **3**(3), 249–265.
Lopez, V., Unger, C., Cimiano, P. & Motta, E. 2013. Evaluating question answering over linked data. *Journal of Web Semantics* **21**, 3–13.
Marginean, A. 2017. Question answering over biomedical linked data with grammatical framework. *Semantic Web* **8**(4), 565–580.
Mazzeo, G. M. & Zaniolo, C. 2016a. Answering controlled natural language questions on RDF knowledge bases. In *EDBT,* 608–611.
Mazzeo, G. M. & Zaniolo, C. 2016b. CANaLI: a system for answering controlled natural language questions on RDF knowledge bases, *Technical Report,* No. 160004.
Mishra, A. & Jain, S. K. 2016. A survey on question answering systems with classification. *Journal of King Saud University-Computer and Information Sciences* **28**(3), 345–361.
Park, S., Kwon, S., Kim, B. & Lee, G. G. 2015. ISOFT at QALD-5: hybrid question answering system over linked data and text data. In *CLEF* (Working Notes).
Pradel, C., Haemmerlé, O. & Hernandez, N. 2012. Swip, a semantic web interface using patterns. In *GKR 2011,* Croitoru, M., Rudolph, S., Wilson, N., Howse, J. & Corby, O. (eds), LNCS 7205, 172–187. Springer.
Pradel, C., Peyet, G., Haemmerlé, O. & Hernandez, N. 2013. Swip at qald-3: results, criticisms and lesson learned. In *Working Notes for CLEF 2013 Conference*.
Radoev, N., Zouaq, A., Tremblay, M. & Gagnon, M. 2018. A language adaptive method for question answering on French and English. In *Semantic Web Evaluation Challenge,* 98–113. Springer.
Ruseti, S., Mirea, A., Rebedea, T. & Trausan-Matu, S. 2015. QAnswer-enhanced entity matching for question answering over linked data. In *CLEF* (Working Notes), 28–35.
Sasikumar, U. & Sindhu, L. 2014. A survey of natural language question answering system. *International Journal of Computer Applications* **108**(15), 975–8887.
Shizhu, H., Yuanzhe, Z., Kang, L. & Jun, Z. 2014. CASIA@V2: a MLN-based question answering system over linked data. In *CLEF 2014 Working Notes Papers.*
Sorokin, D. & Gurevych, I. 2017. End-to-end representation learning for question answering with weak supervision. In *Semantic Web Evaluation Challenge,* 70–83. Springer.
Tran, P. N. & Nguyen, D. T. 2016. A linked data driven semantic model for interpreting English queries in question answering system. In *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication,* 1–5.
Wendt, M., Gerlach, M. & Düwiger, H. 2012. Linguistic modeling of linked open data for question answering. In *Proceedings of Interacting with Linked Data (ILD 2012), Workshop Co-Located with the 9th Extended Semantic Web Conference,* Unger, C., Cimiano, P., Lopez, V., Motta, E., Buitelaar, P. & Cyganiak, R. (eds), May 28, 2012, Heraklion, Greece, 75–87.
Yahya, M., Berberich, K., Elbassuoni, S., Ramanath, M., Tresp, V. & Weikum, G. 2012. Deep answers for naturally asked questions on the web of data. In *Proceedings of the 21st International Conference on World Wide Web,* 445–449.

Xu, K., Zhang, S., Feng, Y. & Zhao, D. 2014. Answering natural language questions via phrasal semantic parsing. In *CCF International Conference on Natural Language Processing and Chinese Computing,* 333–344. Springer.

Zafar, H., Dubey, M., Lehmann, J. & Demidova, E. 2020. IQA: interactive query construction in semantic question answering systems. *Journal of Web Semantics* **64**, 100586.

Zou, L., Huang, R., Wang, H., Yu, J. X., He, W. & Zhao, D. 2014. Natural language question answering over RDF: a graph data driven approach. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data,* 313–324.