Procesamiento del Lenguaje Natural, Revista nº 59, septiembre de 2017, pág□V□□ recibido 31-03-2017 revisado 15-05-2017 aceptado 29-05-2017

ScoQAS: A Semantic-based Closed and Open Domain Question Answering System

Un sistema de búsqueda de respuestas en dominios cerrados y abiertos basado en semántica

Majid Latifi, Horacio Rodríguez, Miquel Sànchez-Marrè
Dep. of Computer Science
UPC University
Campus Nord, C/Jordi Girona, 08034, Barcelona, Spain
{mlatifi, horacio, miquel}@cs.upc.edu

Abstract: Question Answering (QA) has reappeared in research activities and in companies over the past years. We present an architecture of Semantic-based closed and open domain Question Answering System (*ScoQAS*) over ontology resources (not free text) with two different prototyping: Ontology-based closed domain and an open domain under Linked Open Data (LOD) resource. Both scenarios are presented, discussed and evaluated.
Keywords: Semantic question answering, natural language processing (NLP), ontology, linked open data (LOD), linked data (LD)

Resumen: La búsqueda de la respuesta ha reaparecido con fuerza en los últimos años, tanto a nivel industrial como académico. Presentamos una arquitectura de búsqueda de respuesta, *ScoQAS*, basada en la semántica aplicable tanto a dominio cerrado (definido por una ontología) como a dominio abierto, dirigido a repositarios de Linked Open Data (LOD). Los dos se presentan, discuten y son evaluados.
Palabras clave: Respuesta de pregunta semántica, procesamiento del lenguaje natural (PNL), ontología, linked open data (LOD), linked data (LD)

1 Introduction

Currently search engine models for information access break down for more complex information needs. On the one hand, search engines perform keyword search and could not handle natural language questions due to the answer to a question is assumed to be a single web page. On the other hand, major advances in the field of Question Answering (QA) are yet to be realized. Today we are witnessing a large volume of Resource Description Framework (RDF) data which have been published as Linked Data (LD)[1] and on the rise as well.

A QA system obtains its answers by querying over unstructured data or structured information (usually a knowledge base). More commonly, QA systems can pull answers from collection of natural language documents containing free text. Current web that consists of documents and the links between documents is extended by linked data. DBpedia is one of the central LD datasets in linked open data (LOD[2]) project (Bizer, Heath, and Berners-Lee, 2009). Recently, researchers and social analysis companies have more interests on QA systems over LOD.

Question answering researchers are striving to deal with a wide range of question types including: fact, list, definition, opinion, hypothetical, semantically constrained, and cross-lingual questions. Most research in QA focuses on factual QA, where we can distinguish between Wh-queries (who, where, what, how many, etc.), commands (list all, give me, etc.) requiring an element or list of elements as an answer, or affirmation / negation questions. Most difficult kinds of factual questions include those that ask for opinion, like Why or How questions, which require understanding of causality or instrumental relations, and What questions which provide little constraint in the answer type.

[1] https://www.w3.org/standards/semanticweb/data

[2] http://lod-cloud.net/

ISSN 1135-5948

1.1 Motivation of QA Systems

QA systems have been used in multiple scenarios, with increasing number and extended scope of their applications.

Social information seeking is often materialized in online websites such as Yahoo! Answers[3], Answerbag[4], WikiAnswers[5] and Twitter[6]. Another area of success is clinical natural language processing. Regarding the growth of biomedical information, there is a growing need for question answering systems that can help users better utilize the ever-gathering information.

1.2 Summary of ScoQAS

In this work we analyzed the effectiveness of natural language processing (NLP) techniques, query mapping, and answer inferencing both in Closed (1^{st} scenario) and Open (2^{nd} scenario) domains. We focused on the challenges of semantic question answering systems in question interpretation and answer extraction. In *ScoQAS*, we address the deployment of the NLP and artificial intelligence techniques to classify questions with integrating syntactic and semantic parsing using lexical meaning. We exploit an empirical technique that significantly improves the performance of graph-based semantic inference to extract precise answer from the ontology-based domain in the 1^{st} scenario. The technical know-how of mapping method to generate SPARQL query from tuple and its constraints is presented in 2^{nd} scenario.

After this introduction, the organization of the paper is as follows: Section 2 gives a summary of related works. Section 3 contains the general architecture of *ScoQAS*. In Section 4 we describe the closed-domain approach of the *ScoQAS* architecture. Section 5 presents the open domain scenario. Section 6 provides an empirical evaluation in both scenarios. Finally, Section 7 presents contributions, conclusions, and future work.

2 Related Work

QA has been widely studied since the first TREC Question Answering Track in 1999. QA systems have evolved in recent years but still remain anchored on a typical architecture involving question analysis, document or linked data retrieval and, lastly, answer selection strategies. The contribution of each of these steps needs to be evaluated separately in order to understand their impact on the final performance of the QA system. Current trends follow two complementary (or perhaps contradictory) directions:

- Going beyond simple factual QA
- Constraining the search space for the answer by moving to Domain Restricted QA (DRQA) or to systems looking for the answer in structured repositories as ontologies or LOD datasets (or federations of them) as question answering over linked data.

Some of the most relevant systems are independent or open-domain, as QuestIO (Tablan, Damljanovic, and Bontcheva, 2008), AquaLog (Lopez et al., 2007), DeepQA (Kalyanpur et al., 2012)(Ferrucci et al., 2010), QAKiS (Cabrio et al., 2012) while other models are dependent or Closed-domain, as QACID (Ferrández et al., 2009), ONLI+ (Mithun, Kosseim, and Haarslev, 2007) and Pythia (Unger and Cimiano, 2011). Beyond this categorization, in PANTO (Wang et al., 2007), AquaLog (Lopez et al., 2007), DEQA (Lehmann et al., 2012), and QuestIO (Tablan, Damljanovic, and Bontcheva, 2008) are systems that act as natural language interfaces, introducing frameworks, tools and using combined techniques for information retrieval or text mining. However, many of these frameworks or tools do not produce a human-like answer, but rather employ "shallow" methods (keyword-based, templates, etc.) to produce a list of documents or excerpts of documents containing the likely answer highlighted.

3 The ScoQAS Architecture

In Figure 1, the general architecture of *ScoQAS* is depicted and the initial model was presented in 2013 (Latifi and Sanchez-Marre, 2013). The *ScoQAS* employs NLP techniques and combines tuple pattern with NSIF[7] for question classification. In question interpretation phase, it uses a heuristic method for constraining the interpretations of the questions with semantic tagging to generate the question graph to facilitate the complexity of

[3]https://answers.yahoo.com/
[4]http://www.answerbag.com/
[5]http://www.answers.com/
[6]https://twitter.com/
[7]NLP Semantic-based Interchange Format

the inference algorithms. More technical issues are explained in Section 4.

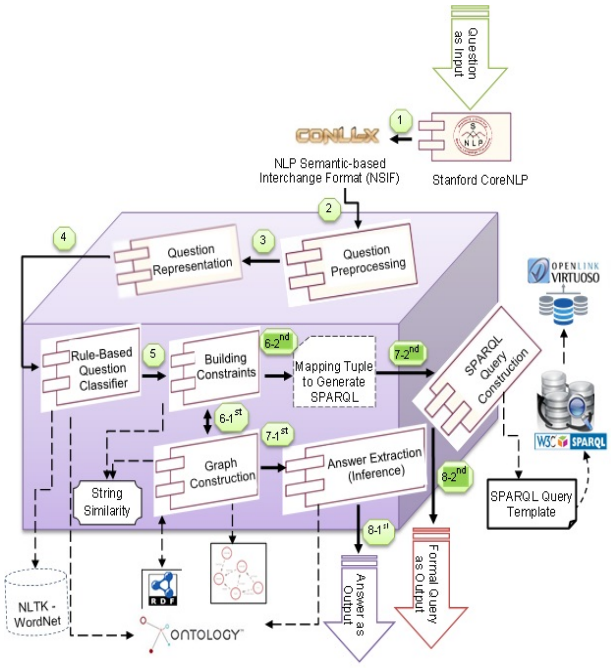

Figure 1: Architecture of Semantic-based closed and open domain Question Answering System (ScoQAS)

ScoQAS performs over ontologies and operates on two scenarios. The purpose of these scenarios is to sketch out the specific conduct in two domains with the aim of consolidating the pros and cons for designing and implementing the integrated approach in order to demonstrate the adaptabilities of our approaches to achieve the considerable results. The first scenario is Closed-domain QA system, where the domain is restricted by an ontology, and the second one is an Open-domain QA system where the answers are retrieved from a LOD knowledge base. In the 1^{st} scenario the possible instantiations are reduced to changing the supported ontology, while in the 2^{nd} scenario what changes are the involved LOD datasets.

As shown in Figure 1, there are specific components for each of them and common ones usable in both scenarios. Those of the components which are outside of the cube are external tools reused by *ScoQAS* such as Stanford CoreNLP parser, WordNet, and SPARQL query endpoint[8], etc. The components inside of the cube (as question preprocessing, question representation, rule-based question classifier, building constraints, SPARQL query generation, graph construction, and answer inference) were developed in ScoQAS.

The question processing phase, common to both scenarios, aims to classify a question in order to bind Question Type (QT) and determine the Expected Answer Type (EAT) and extract further constraining information. In the 1^{st} scenario, the graph construction is used to generate the question graph with specific format. Instead of it, in the 2^{nd} scenario, other specific components are applied to generate SPARQL format to deal with the mapping challenges. Another pair of specific components is related to the answer extraction task where the 1^{st} scenario uses a heuristic inference mechanism over a graph extracted from the ontology, while the 2^{nd} scenario generates SPARQL query template over the LOD knowledge bases.

As shown in Figure 1, the common steps in both scenarios includes 1, 2, 3, 4, and 5. In the 1^{st} scenario, the implementation of the ontology traversing approach with constraint variables (6-1^{st} step), building of specific graph format (7-1^{st} step), the answer extraction task (8-1^{st} step) are specific components. In contrast, the dedicated steps for 2^{nd} scenario are 6-2^{nd}, 7-2^{nd}, and 8-2^{nd}. In Sections 4 and 5, we describe how the common and specific components are exploited in each step of its scenario.

4 Closed Domain QA

The basic idea behind the closed domain (1^{st} scenario) is to devise an inference mechanism performing over a question graph (QGraph), built from the NSIF representation enriched with ontology information. We used improved Enterprise ontology as knowledge base in this scenario (Latifi, Khotanlou, and Latifi, 2011). There are several challenges which should be addressed:

- Building the NSIF representation for each QT.
- Exploitation of the graph representation to demonstrate the semantic relationships between words in the question and the corresponding nodes in the ontology.
- Building inference engine to extract answer(s) from the graph produced during question processing.

[8]https://virtuoso.openlinksw.com/

The ways of facing theses issues is described in the following sections.

4.1 Question Preprocessing

To achieve the aims of the desired semantic-based QA system, annotating and finding out the structure of the question syntactically and semantically is the significant step through doing NLP task. In this regards, we present a Semantic-based Interchange Format by relying on NLP techniques (NSIF) for representing information extracted from the question and, if available, from the ontology. The primary information of the NSIF consists of tokenization and morphological analysis such as lemmatization, POS tagging, and named entity recognition (NER). The NSIF is using dependency parsing information in order to complete the whole syntactical information of the question. In addition, the NSIF is generated in order to exploit it in downstream processes as an enriched representation of the question in the mapping or in the answer retrieval process. In the first step of both scenarios, the basic information of NSIF is extracted from Stanford CoreNLP[9], e.g., the basic dependency parsing information of Example 1 (Q1) is configured in the NSIF format (see Figure 2).

(1) *"Where is the manager of ITC working in the organization?"*

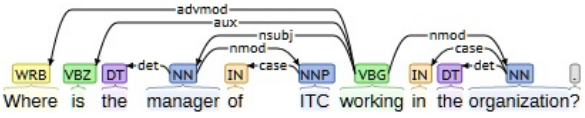

Figure 2: Basic dependencies provided by Stanford CoreNLP parser for question Q1

4.2 Question Representation

ScoQAS should be able to access and control all of the extracted data from NSIF format and preprocessing step in order to represent the syntactic structure of question and other information gathered from ontology. Its task is to bind up the elements of NSIF into semantic information such as related terms in ontology items corresponding to the token.

4.3 Rule-Based Question Classifier

The key point of Question Classifier (QC) is analyzing the question to a degree that allows determining the question type, QT, of the query (from a tagset of 75 QTs, see Table 1 for some examples) and deriving from it the the Expected Answer Type (EAT) for the answer. The aim is that this classification, potentially with other constraints on the answer, will be used by a downstream process for selecting the correct answer from a set of candidates. Additionally to the classification task, this module extracts all the information necessary for the rest of the QA process. The QT "Where_Person_Action" is assigned to the question Q1 (see example 1), i.e. we are looking for a place where a person carried out some action.

Each QT rule consists of a set of Conditions and Actions rules. When all of the Conditions for question are satisfied then Actions rules are executed. The rule conditions of the example can be paraphrased as following: If the token "Where" starts the sentence and there is either a token being a Named Entity (NE) of class PERSON or a token able to be mapped into a node in the ontology being a subclass of PERSON, and there is a token in the question being a verb or a verbal nominalization then the QT "Where_Person_Action" is extracted.

ID	Question	QT	Scenario
1	Where is the manager of ITC working in the organization?	Where_Person_Action	1st
2	How much is the insurance premium deductions for Ali?	Howmuch_Properties_Person	1st
3	Give me all female Russian astronauts.	Who_Properties	2nd
4	When was the Statue of Liberty built?	When_Action_Compound-Properties	2nd
5	In which country is the Limerick Lake?	Where_Properties_GEO	2nd

Table 1: Examples of QTs in both scenarios

4.4 Generating the Constraints

From the QT, and using the information placed into the NSIF, a ranked collection of possible EAT can be inferred. For example, for Q1 the most likely EAT is simply a Location, because of the "where" token, but another option, ranked below, could be whatever part of an ORGANIZATION because in this case a COMPANY (subclass of ORGANIZATION), named ITC, is mentioned. Constructing such constraints provide more information about the nature of questions

[9]http://stanfordnlp.github.io/CoreNLP/

and help to get a precise answer. The constraint's units include constraints and variables acting the former as restrictors over the values of variables. The constraints hold syntactic or semantic relations between the question keywords and its produced QT's.

The constraints can be classified as mandatory constraints (MC) that have to be satisfied by the answer, and optional constraints (OC) which simply increases the answer credibility score when satisfied. The QT defines the set of mandatory constraints for a question. For instance, Q1 (see example 1), where the QT is "Where_Person_Action", both the 'Person' and the 'Action' should be constrained in the target space (the ontology in this case, the LOD repository in the 2^{nd} scenario). The nodes mapped to the tokens in the query corresponding to the Person (the manager of ITC) and the Action (working) should be constrained by the relation holding between them (subject).

Generation of MC is placed within the action part of the rules (if the conditions are satisfied then MC is generated). MC depends basically on the QT and is derived from the mentions associated with the variables of the QT and their dependency relations. For instance, for Q1 the QT has two parameters, the person, and the action involved. In this case, two variables X1 and X2 are introduced and the corresponding mentions are placed into MC: tk_PER(3, X1) and tk_ACTION(6, X2). From the path between the tokens 3 and 6 in the dependency tree we can include in MC the predicate nsubj(X1,X2) (see Figure 2). In the 1^{st} scenario, more entities (variables) and relations can be extracted from the ontology and placed into MC once the question graph (QGraph) has been built as shown in Section 4.5. In this example "ITC" (token 5) is found in the ontology as an instance of class COMPANY and also "organization" (token 9) is found as a class, so, tk_ONTOLOGY(9,X3) and tk_ONTOLOGY(5,X4), isa(COMPANY, ORGANIZATION), instance(X4, COMPANY), class(X3, ORGANIZATION). As MC has grown, a new iteration on the dependency tree is attempted, in this case adding the prep_IN(X2, X3), and so on.

4.5 Generating the Graph

The ontology provides the semantic space where answers can be found and extracted. Some variables have already being mapped into ontology entities (classes, slots, instances) during the QC and building constraint steps (as was the case of 'manager', 'ITC', and 'organization' in the Q1 example). So the QGraph is created. We can define the QGraph as a subgraph of the virtual graph representing the ontology. The QGraph is used both as a search space for locating the answer and as a resource for enriching the constraint sets. Its context is analyzed to find the relations between the variables, arguments, ontology classes, ontology instance corresponding to the variables, and EAT classes and EAT Instances.

During the generation of the QGraph for QT, its context is evolving, so the general procedure is as follows:

1. Extracting the keywords of the question (in the Q1, "manager","ITC","working","organization"), enrich this set with morphological variants and WN synonyms.
2. Looking at the ontology dictionaries performing approximate matching in a way that scored matches are obtained between keywords and ontology items.
3. Building the QGraph using the instances and classes obtained before as nodes and slots as edges. Both nodes and edges are weighted with the confidence scores got from the approximate matches.
4. Expanding QGraph with paths extracted from the ontology trying to link as many nodes as possible.

The portion of QGraph that keeps the Q1 information is depicted in Table 2. It demonstrates the achieved information from initial MC as graph format.

4.6 Inference to Elicit Exact Answer from Graph Format

In the 1^{st} scenario, the EAT is a set of classes belongs to the ontology (in fact nodes of the QGraph). The answer has to be an instance of one of these classes. The searching process consists of navigating over the QGraph looking for nodes X satisfying the constraints is shown as pseudo code in Figure 4.

5 Open Domain QA

In the case of the *2^{nd} scenario*, the first five steps, i.e. the common components, are the

Node	Edge: Label	Var
X1	-	X1
3	-	50
X2	-	X2
6	-	100
-	('X1', 3): tk_PER	-
-	('X2', 6): tk_ACT	-
-	('X1', 'X2'): nsubj	-
X5	-	X5
Manager	-	250
X8	-	X8
manager_title	-	400
-	('X5', 'Manager'): class_PER_0	-
-	('X1', 'X5'): ont_PER_0	-
-	('X8', 'manager_title'): slot_PER_1	-
-	('X5', 'X8'): Slot_1	-

Table 2: QGraph nodes and edges with matched ontology items (Var: Variable)

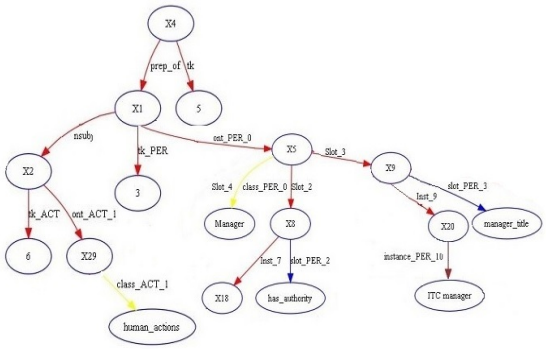

Figure 3: A part of produced QGraph for Q1

same as 1^{st} scenario so that the constraint set is produced. As no domain ontology exists, the constraints are only those generated from NSIF are used in order to interpret the question. To deal with the issue of mapping natural language to SPARQL query format, we have implemented a method to generate SPARQL query associated to NSIF, MC, and OC.

As no constraints regarding domain ontology exist in this scenario, the set of MC uses to be smaller and, so, as the search is less constrained, the ambiguity of the answer candidates is higher. Hopefully the complexity of the questions (at least within QALD contests) use to be smaller too.

5.1 Pre-processing Steps

The objective of this module is to map natural language questions, previously processed by the QC module (obtained MC and NSIF), into SPARQL queries. We have performed two following pre-processing steps:

1. Instance(X,Y), member(Y,EAT)
2. For all Z, so that member(Z,MC.nodes), connected(X,Z)
3. If no X is found, there is no answer
4. If only one X is found, it is as answer
5. If more than one X is found, the most likely one, from the scores of all the paths from X to all the Z is the answer.

Figure 4: Searching process in QGraph

A) General Pre-processing: We have pre-indexed all the Yago classes, DBpedia(classes, properties). Besides, we have built an index for all the simple word forms contained in the previously indexed multi-word entries. For instance, from the property http://dbpedia.org/property/u.s.SeniorNationalTeamMember, the set 'u.s.', 'Senior', 'National', 'Team', 'Member' has been extracted.

B) Dataset Pre-processing: We collect all the actions occurring in the question dataset (all the tokens referred within the constraint set as 'tk_ACT'). For each action we obtain its lemma, the set of all its variants and the set of forms derived from these ones (using NLTK's WN tools). For all the classes corresponding to EAT categories, both generic or specific, i.e., those referred within the constraint set under the key 'tk_Type', we collect the set of upper classes in Yago and DBpedia ontologies, using the indexes produced in the previous step. e.g., in Example 2 (Q2) new terms as "Russia", "astronaut", "cosmonaut", etc. are generated for improving the recall when looking at DBPedia.

(2) *"Give me all female Russian astronauts."*

5.2 Mapping Tuple Information to Construct SPARQL Query

Here, the goal is to construct SPARQL queries for a given set of constraints variables that will be next sent to the Virtuoso DBpedia endpoint in order to get the final answer. We generate queries using all bounded variables to corresponding QT, EAT, MC which have been indexed by symbols (see Table 3).

Let us to consider the details of the Q2:

QT: Who_Properties

MC: {tk_Quant: 0 ('all'), tk_Props: [1, 2] ('female','Russian'), tk_Type: 3 ('astro-

MC Label	MC Description
tk_PER	The token(s) indicate person entity
tk_GEO	The geographic token(s) occurring in the question
tk_ACT	The action token(s) like as verb
tk_Type	The token(s) indicate th type of EAT
tk_Quant	The token(s) show the quantifier
tk_Props	Set of independent constraints tokens

Table 3: Some MC symbols with its concepts

nauts')}

First EAT is set to 'astronaut' using WN lemmatizer. Then the set of keywords, including tk_Props and tk_Type is expanded using WN NLTK tools, resulting in {'Russians', 'astronauts', 'Astronaut', 'female',.., 'Female', 'cosmonauts'}. A lot of classes, properties, and instances are found, e.g., for 'Russian', 531 Yago classes and 1 DBpedia class are found. For 'Astronaut', 2 Yago classes, 2 DBpedia properties, and 1 DBpedia class are found. As the QT "Who_Properties" deals with satisfying set of independent constraints to find person(s), we tried to collect Yago or DBpedia classes that could be related with our target by means of rdf:type relation. We try to select classes covering at least two of the three keywords, so we obtained the set {http://dbpedia../yago/FemaleAstronauts, http://dbpedia../yago/RussianCosmonauts}. These classes covers two of keywords and the conjunction of both covers the three keywords. The SPARQL query is built:

Select DISTINCT ?x WHERE {
?x rdf:type http://dbpedia.org/class/yago/RussianCosmonauts
?x rdf:type http://dbpedia.org/class/yago/FemaleAstronauts }

6 Evaluation

The ScoQAS is evaluated in the two scenarios. As there is lack of golden data for the 1^{st} scenario we focus on quantitative evaluation while the 2^{nd} scenario is mainly qualitative. There is a set of dimensions in order to analyze the efficiency of the QA system which has a negative/positive impact on the run time and the accuracy. One of the major challenges of evaluation of the 1^{st} scenario is a lack of predefined benchmark(s). Therefore, we defined measures that demonstrate the actual complexity of the problem and the actual efficiency of the solutions. Hence, a baseline model is determined to evaluate the accuracy of the ScoQAS. We analyzed the 1^{st} scenario based on six steps as shown in Table 4.

Processed	Correct Parsed	Correct QC	Correct EAT	Correct Constraint	Correct QGraph	Correct Ans. Infer.	Correct Ans.	Global Accuracy
18	18	18	17	16	7	6	6	**0.33**

Table 4: Global accuracy of 1^{st} scenario, (Ans.: Answer, Infer.: Inference)

In 2^{nd} scenario, with respect to other semantic QA system evaluation benchmarks, we use a series of evaluation campaigns on QALD[10]. For developing this approach, set of QALD-3 test set is used as a training set. We analyzed open domain QA system based on four dimensions consisting of QC, question constraints, EAT, and mapping question to formal query (SPARQL). The results of the system on the training and the test set are presented in Table 5.

QALD	Pr	AO	C	R	P	F-M
QALD-2 Test	99	81	35	0.82	0.43	0.5642
QALD-3 Training	100	82	31	0.82	0.38	0.5193
QALD-4 Test	50	36	28	0.72	0.78	0.7488
QALD-5 Test	59	48	25	0.80	0.52	0.6303
ScoQAS Average	**-**	**-**	**-**	**0.79**	**0.527**	**0.6156**

Table 5: Evaluation of ScoQAS over QALD benchmarks. Processed (Pr), Answer Obtained (AO), Correct (C), Precision (P), Recall(R) and F-Measure (F-M)

The ScoQAS is compared to the gold standard with respect to precision and recall for QALD winner and median (see Table 6).

QALD	Median F-M	Top F-M
QALD-2	0.38	0.46
QALD-3	0.36	0.90
QALD-4	0.36	0.72
QALD-5	0.40	0.73

Table 6: The QALD competitions results in F-Measure (F-M)

[10]http://qald.sebastianwalter.org/

7 Conclusion and Future Works

The empirical evaluation shows the effectiveness and scalability of *ScoQAS*. We employed AI and NLP techniques to interpret questions semantically, classification and make graph-based inference. The novel method in building constraints were presented to formulate the related terms in syntactic-semantic aspects using Semantic Web technologies. This innovation helps to make a question graph which facilitate to infer for getting an exact answer in the closed domain. The presented approach provides a convenient method to generate SPARQL query template to crawl in the LOD resources in the open domain.

The research findings show that, using statistical techniques in NLP is really promising particularly in terms of recall. The future work is open to apply statistical features in some of the processes, e.g. question classification and inference, in order to increase the accuracy and efficiency of the *ScoQAS*.

Acknowledgments

We are grateful for the suggestions from three anonymous reviewers. Dr. Rodríguez has been partially funded by Spanish project "GraphMed" (TIN2016-77820-C3-3R). This work has been partially funded by the Spanish Thematic Network "Diversificación en Aprendizaje Máquina y Aplicaciones" (DAMA), under grant code TIN2015-70308-REDT (MINECO/FEDER EU).

References

Bizer, C., T. Heath, and T. Berners-Lee. 2009. Linked data-the story so far. *IJSWIS*, 5(3):1–22.

Cabrio, E., J. Cojan, A. Palmero Aprosio, B. Magnini, A. Lavelli, and F. Gandon. 2012. QAKiS: an open domain QA system based on relational patterns. In *Int. Conf. Posters Demonstr. Track-Volume 914*, pages 9–12.

Ferrández, Ó., R. Izquierdo, S. Ferrández, and J. L. Vicedo. 2009. Addressing ontology-based question answering with collections of user queries. *IPM*, 45(2):175–188.

Ferrucci, D., E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty. 2010. Building Watson: An Overview of the DeepQA Project. *AI Mag.*, 31(3):59–79.

Kalyanpur, A., B. K. Boguraev, S. Patwardhan, J. W. Murdock, A. Lally, C. Welty, J. M. Prager, B. Coppola, A. Fokoue-Nkoutche, L. Zhang, Y. Pan, and Z. M. Qiu. 2012. Structured data and inference in DeepQA. *IBM J. Res. Dev.*, 56(3.4):351–364.

Latifi, M., H. Khotanlou, and H. Latifi. 2011. An efficient approach based on ontology to optimize the organizational knowledge base management for advanced queries service. In *IEEE 3rd ICCSN*, pages 269–273.

Latifi, M. and M. Sanchez-Marre. 2013. The Use of NLP Interchange Format for Question Answering in Organizations. In *IOS Press. Front. Artif. Intell. Appl.*, pages 235–244.

Lehmann, J., T. Furche, G. Grasso, A.-C. N. Ngomo, C. Schallhart, A. Sellers, C. Unger, L. Bühmann, D. Gerber, K. Höffner, D. Liu, and S. Auer. 2012. DEQA: Deep web extraction for question answering. In *ISWC 2012*, volume 7650 of *LNCS*, pages 131–147. Springer Berlin Heidelberg, nov.

Lopez, V., V. Uren, E. Motta, and M. Pasin. 2007. AquaLog: An ontology-driven question answering system for organizational semantic intranets. *Web Semant.*, 5(2):72–105.

Mithun, S., L. Kosseim, and V. Haarslev. 2007. Resolving quantifier and number restriction to question OWL ontologies. In *3rd SKG 2007*, pages 218–223.

Tablan, V., D. Damljanovic, and K. Bontcheva. 2008. A natural language query interface to structured information. In *LNCS*, volume 5021 LNCS, pages 361–375.

Unger, C. and P. Cimiano. 2011. Pythia: Compositional meaning construction for ontology-based question answering on the semantic web. In *LNCS*, volume 6716 LNCS, pages 153–160.

Wang, C., M. Xiong, Q. Zhou, and Y. Yu. 2007. PANTO: A Portable Natural Language Interface to Ontologies. *Eswc*, 4519:473–487.