Received: 11 June 2018 | Revised: 1 July 2019 | Accepted: 29 July 2019

DOI: 10.4218/etrij.2018-0312

ORIGINAL ARTICLE

ETRI Journal WILEY

Ontology-lexicon–based question answering over linked data

Mehdi Jabalameli | **Mohammadali Nematbakhsh** | **Ahmad Zaeri**

Department of Software Engineering, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran

Correspondence
Mohammadali Nematbakhsh, Department of Software Engineering, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.
Email: nematbakhsh@eng.ui.ac.ir

Recently, Linked Open Data has become a large set of knowledge bases. Therefore, the need to query Linked Data using question answering (QA) techniques has attracted the attention of many researchers. A QA system translates natural language questions into structured queries, such as SPARQL queries, to be executed over Linked Data. The two main challenges in such systems are lexical and semantic gaps. A lexical gap refers to the difference between the vocabularies used in an input question and those used in the knowledge base. A semantic gap refers to the difference between expressed information needs and the representation of the knowledge base. In this paper, we present a novel method using an ontology lexicon and dependency parse trees to overcome lexical and semantic gaps. The proposed technique is evaluated on the QALD-5 benchmark and exhibits promising results.

KEYWORDS

linked data, ontology lexicon, question answering, semantic web

1 | INTRODUCTION

Many knowledge bases have been published in Linked Open Data (LOD). LOD currently contains hundreds of knowledge bases that are very difficult to access for typical users. Although some formal languages, such as SPARQL, have been introduced to query such knowledge bases, learning these languages is difficult for typical users and they must be familiar with the structures of knowledge bases. Question answering (QA) systems in the form of natural language (NL) interfaces have been proposed to solve this problem. A QA system takes user information requests in the form of NL and extracts exact results from a knowledge base. Most QA systems convert NL questions into structured queries or formal representations [1,2]. One of the main problems in this conversion process is the difference between the words or phrases used by users and the vocabulary used in the knowledge base. This difference is known as a lexical gap. As an example, consider the following question:

Who is playing in Spanish movies produced by David Lynch?

To answer this question, the following SPARQL query can be executed over a knowledge base, such as DBpedia (we abbreviate URI namespaces with common prefixes, see http://prefix.cc for details):

```
SELECT ?who WHERE {
  ?movie   dbo:starring    ?who.
  ?movie   rdf:type        dbo:Film.
  ?movie   dbo:language    res:Spanish_language.
  ?movie   dbo:producer    res:David_Lynch. }.
```

In the example above, the word "movie" is used in the NL question, whereas in the SPARQL query, the corresponding property "*dbo:Film*" is used.

Another problem in the conversion of NL questions into the SPARQL is the difference between the manner in which users expresses their need and how that knowledge is represented in the knowledge base. This type of difference is

known as a semantic gap. In the example above, the expression "Spanish movie" could relate to a resource with the type "*Spanish_Movie*," or to a resource with the type "*movie*" or "*film*" with another property, such as "*dbo:language*," with the value "*res:Spanish_language*."

In this paper, we propose a QA system that uses an ontology lexicon and the dependency parse trees of NL questions to overcome lexical and semantic gaps. An ontology lexicon contains information regarding how the elements of an ontology can be verbalized in a given language. For example, consider the following entry in an ontology lexicon in the Lemon design patterns format [3]:

StateVerb("play", dbo:starring,
propSubj = PrepositionalObject ("in"), propObj = Subject).

This entry states that the verb "*play*" in a NL sentence may relate to the property "*dbo:starring*," where the subject of the property is the prepositional object of the sentence and the object of the property is the subject of the sentence. Based on this information and the dependency parse tree of the input question, we can map the verb of the input question to the correct property to preserve the order of its subject and object.

By using the dependency parse tree of an input question with an ontology lexicon, we can capture different mappings between input questions and a knowledge base. This task is accomplished by using rules extracted from training questions. All the SPARQL queries obtained from question mappings are ranked and the best SPARQL query with a nonempty answer is returned as the final result.

In some previous studies, ontology lexicons have been used as lookup tables to map the words in input questions to the concepts in the knowledge base. Lexicons have also been used as bases to construct grammar for parsing input questions. Both of these methods have drawbacks. Using an ontology lexicon as a lookup table does not exploit all available detail in the entries of the ontology lexicon. For example, the verb "influence" can be mapped to both the "*dbo:influenced*" and "*dbo:influencedBy*" properties, even though the orders of the subject and object in these properties are different. Ignoring such details could produce an incorrect SPARQL query. Additionally, grammar constructed from an ontology lexicon is very sensitive to input questions and may fail to parse input questions when minor changes occur in those questions. We have attempted to resolve these issues by generating correct and feasible SPARQL queries. These queries can be executed directly over a knowledge base or optimized using existing methods [4,5].

The main contributions of this paper are as follows:

- A novel QA approach is proposed using a general-purpose dependency parser and ontology lexicon in a simple manner.
- The proposed QA approach attempts to find all possible interpretations of input questions and ranks them to find the best interpretation.

The remainder of this paper is organized as follows. Some related studies on QA systems are discussed in Section 2. The architecture of the proposed approach is described in Section 3. In Section 4, the effectiveness of the proposed approach is evaluated. Finally, conclusions and future work are discussed.

2 | RELATED WORK

Recently, many studies have been published in the field of QA. Based on formal grammars, in ORAKEL [6] and Pythia [7], the semantics of a question can be obtained by combining the meanings of different parts of the question. The main drawback of these systems is that if a question cannot be parsed by a specified grammar, the system will fail. In our approach, if a portion of an input question cannot be processed, it is simply ignored and the system attempts to find an answer based on the remainder of the question.

To convert an input question into a SPARQL query, TBSL [8], QUINT [9], GeoQA [10], LODQA [11], and the system proposed by Biermann and his colleagues [12] all utilize template-based approaches. In such approaches, templates are generated based on linguistic analysis of input questions. A template is instantiated by mapping the expressions in an input question to the elements of a knowledge base. Although such approaches can process complex questions, constructing templates for all variations of input questions is very difficult.

Some other systems consider QA as a semantic parsing problem and attempt to train a semantic parser to map input questions to appropriate logical forms to be executed over a knowledge base [13,14]. Such systems require significant training data that are time consuming and labor intensive to create.

The main focus of other studies has been resolving ambiguities arising from the mapping of phrases in input questions and semantic items in knowledge bases. DEANNA [15] formulates the QA task as an integer linear programming (ILP) problem and resolves such ambiguities by solving the ILP problem. In CASIA [16], ambiguities are modeled as soft constraints in Markov logic networks. SINA [17] employs a hidden Markov model (HMM) and utilizes the optimal path from the HMM for disambiguation. In our method, disambiguation is performed in the final step by ranking generated SPARQL queries.

Instead of converting an input question into a SPARQL query, in Treo [18], APEQ [19], and the system proposed by Zhu and his colleagues [20], a knowledge base is viewed as a graph and graph exploration algorithms are used to find the answer to a question. In these systems, various knowledge base entity mentions in the input question are identified as pivot points. Based on these pivot points, an exploration algorithm is executed to find answers connected to the points by a path in the graph. Unfortunately, if answers are not connected

to the pivot points by a direct path or if a question involves complex operations, such as aggregation, such systems cannot find an answer.

Some QA systems are limited to a specific domain, such as medicine, agriculture, or geography [10,21–23]. In such systems, there is less ambiguity. Therefore, the accuracy of answers in such systems is typically higher than that in open-domain QA systems. However, it is difficult and costly to extend such systems to new domains.

3 | SYSTEM ARCHITECTURE

In this section, we present the architecture of the proposed approach. As shown in Figure 1, the proposed system consists of six components: syntactical analysis, template Generation, NE recognition, candidate selection, ranking, and answer retrieval.

3.1 | Syntactical analysis

Syntactical analysis takes an NL query as an input and generates part of speech (POS) tags, as well as a dependency parse tree for the input NL query. Currently, POS tagging and dependency parsing tools are used in many NL processing tasks and provide reasonable accuracy. Unlike some previous methods that require a large amount of training data to identify the structure of an input question, we used these general-purpose tools to eliminate the work required to collect large training datasets. The syntactical analysis component uses the Stanford CoreNLP tools [24]. Figure 2 presents the output of this stage.

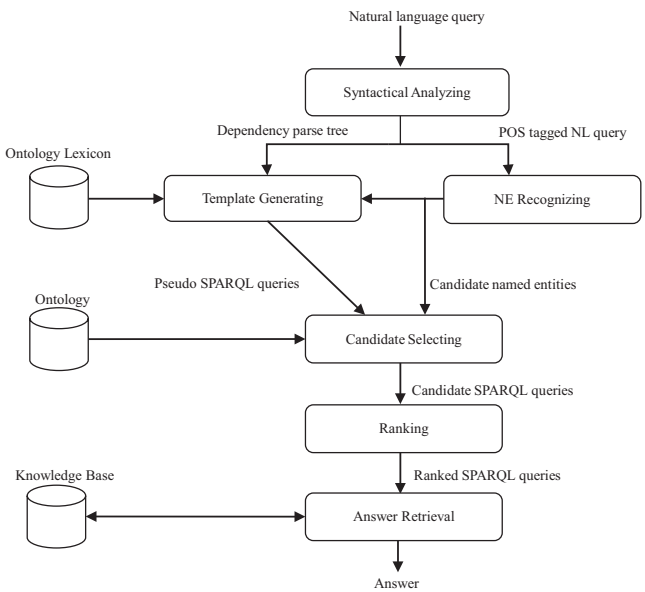

FIGURE 1 Proposed system architecture

3.2 | Named entity recognition

The words and phrases in a question that refer to named entities must be recognized and linked to knowledge base resources. To this end, we used DBpedia Spotlight [25], which detects entity mentions in a query and returns not only the corresponding resources, but also a score for each resource that represents the level of relatedness between the word or phrase and resource.

By using POS tags and some predefined rules, we eliminate certain entity mentions that are unlikely to link to a resource (eg, interrogative words). We retain all other entity mentions. Table 1 presents the final entity mentions and scores for the running example.

3.3 | Template generation

Template generation plays the main role in our approach and consists of two parts: question classification and pseudo-SPARQL generation.

Question classification classifies questions into four categories: imperative (eg, list queries), predicative (eg, "wh-" questions), counting (eg, questions starting with "how many…"), and affirmation (eg, queries whose results are "yes/no"). This classification is performed using certain predefined rules. Based on these rules, the running example is classified as predicative.

Following classification, according to the parse tree and ontology lexicon, various pseudo-SPARQL queries are generated. Pseudo-SPARQL queries are similar to SPARQL queries, but they may contain named entities without specific URIs (eg, "Spain" instead of "*res:Spain*").

Pseudo-SPARQL queries are generated using an algorithm as follows (Figure 3). The algorithm starts at the root of the dependency parse tree and checks the POS tag of the root. For each node in the dependency parse tree, it looks up the corresponding entry for the node label in the ontology lexicon.

To increase the probability of finding entries in the ontology lexicon, we expand node labels using synonyms from the WordNet [26] dictionary All words in all synsets of a node label are searched for in the ontology lexicon.

For each entry found in the ontology lexicon, a new triple is created according to the syntactic behavior and semantics of the entry specified in the ontology lexicon.

In the running example, the dependency parse tree starts with the verb "*playing*." The ontology lexicon may contain the following entry among others:

StateVerb("play", dbo:starring,
propSubj = PrepositionalObject("in"), propObj = Subject)

From this entry, the following triple is generated:

?movie dbo:starring ?who

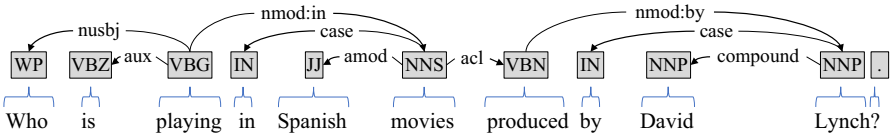

FIGURE 2 POS tags and dependency parse tree of the running example generated using the online Stanford CoreNLP demo

TABLE 1 The output of the named entity recognition step

Phrase	Related entity	Score
David Lynch	*res: David_Lynch*	0.998
David Lynch	*res: David_Lynch_(wine_expert)*	0.00175
David Lynch	*res: David_Lynch_(pioneer)*	1.53e–12
David Lynch	*res: Early life of David Lynch*	1.27e–15
David Lynch	*res: David Lynch (director)*	1.27e–16
David Lynch	*res: David Lynch (politician)*	1.27e–16

Because the running example is a "wh-" question, a pseudo-SPARQL query with the variable "*?who*" as a select variable is generated. By adding the generated triple to the pseudo-SPARQL query, we obtain the following query:

```
SELECT ?who WHERE {
  ?movie   dbo:starring   ?who.}.
```

By recursively processing children of the root node "*starring*," additional triples are generated and are added to the pseudo-SPARQL query. The final pseudo-SPARQL query can be as follows:

```
SELECT ?who WHERE {
  ?movie   dbo:starring   ?who.
  ?movie   rdf:type       dbo:Film.
  ?movie   dbo:language   res:Spanish_language.
  ?movie   dbo:producer   David Lynch. }.
```

```
L = Ontology Lexicon
Ts = SPARQLs Tree
Check(node:DepTreeNode)
{if L.contains(node.label, node.POStag)
  foreach entry in L.getEntries(node.label, node.POStag)
  { t = createTriple(entry , node)
    Ts.expand(t)
    foreach child c of node do
        Check(c)
  }
}
```

FIGURE 3 Template generation algorithm

If more than one triple is generated for a node, each triple is added to the previously generated pseudo-SPARQL query to consider every interpretation of the original question. To retain these interpretations, we store all pseudo-SPARQLs results in a tree. Every branch in this tree represents an interpretation of the query. Finally, all leaves of the tree are returned.

3.4 | Candidate selection

To convert pseudo-SPARQL queries to executable SPARQL queries, all entity mentions in a pseudo-SPARQL query must be replaced with the corresponding resources from the knowledge base. Before replacing entity mentions, we check the type compatibility of the domain and range of the property with the type of the corresponding resource. The domains and ranges of properties can be obtained from the ontology or from the instances of properties in the knowledge base. In the latter case, the type of resource used in the subject (or object) of a property in the knowledge base is considered to be the domain (or range) of that property.

In the final generated pseudo-SPARQL query, the range of the property "*dbo:producer*" and the type of "*res:David_Lynch*" are both "*dbo:Agent*." Therefore, the entity mention of "David Lynch" can be replaced with "*res:David_Lynch*."

3.5 | Ranking

To rank queries, we compute the score of a SPARQL query as the mean score of all triples in that query. The score of a triple is computed as product of the scores of its subject, property, and object. If the subject (or object) of a triple is a resource in the knowledge base that is extracted by DBpedia Spotlight with a score α, then the score of that subject (or object) will be α. Otherwise, the score of the subject (or object) will be 1.

The score of a property is 1 unless it is obtained from the synonyms of a word or phrase. In such cases, we reduce the score by a decay factor β ($0 < \beta < 1$). We experimentally set β to 0.9.

In the final SPARQL query in Section 3.3, if the entity mention "David Lynch" is replaced with "*res:David_Lynch*," which has a final score of 0.998, then the score of the final triple will be $1 \times 1 \times 0.998 = 0.998$. The scores of the other

triples are 1. Therefore, the score for this query is (1 + 1 + 1 + 0.998)/ 4 = 0.9995.

After calculating the scores for the SPARQL queries, we sort the queries in decreasing order by score.

3.6 | Answer retrieval

In the final step, we execute the ranked SPARQL queries over the SPARQL endpoint of the knowledge base to find the query with the highest score and a non-empty result. If such a query is identified, its result is returned as the answer to the question. Otherwise, the system refuses to answer the question.

4 | EVALUATION

The proposed system was evaluated on the QALD-5 benchmark. Question Answering over Linked Data (QALD) is a series of challenges on question answering over Linked Data. It provides relevant benchmarks for evaluating QA systems. QALD-5 is the fifth installment of QALD. The QALD-5 dataset contains 300 questions as a training set and 49 questions as a test set. All questions must be answered using the DBpedia 2014 dataset. We selected the DBpedia ontology lexicon [27] for the proposed system. It covers 98% of the classes and 20% of the properties in the DBpedia ontology with approximately 1.3 entries per class and 2.4 entries per property. Because this lexicon was created manually and did cannot contain all lexicalizations of the ontology concepts required for processing the questions in the benchmark, we manually added some additional entries to the lexicon. These entries are of the same type as those in the original DBpedia ontology lexicon and were generated by reviewing the questions that the system must answer. Therefore, prior to running the entire system, these entries are added to evaluate the performance of various parts of the system. If these entries are not added, the system will fail to operate properly.

To evaluate the effectiveness of our approach, we calculated the recall, precision, and *F*-measure for each question q as follows:

$$\text{Recall}(q) = \frac{\text{Number of correct system answers for } q}{\text{Number of gold standard answers for } q},$$

$$\text{Precision}(q) = \frac{\text{Number of correct system answers for } q}{\text{Number of system answers for } q},$$

$$F\text{-measure}(q) = \frac{2*\text{Recall}(q)*\text{Precision}(q)}{\text{Recall}\,(q)+\text{Precision}\,(q)}.$$

Table 2 lists the results for the QALD-5 test questions and compares the proposed method to four state-of-the-art systems that participated in the QALD-5 challenge. The results for these systems were reported in the QALD-5 overview paper [19]. The table lists the numbers of questions each system could answer (processed column) and the numbers of right and partially right answers (with *F*-measures strictly between 0 and 1). The overall precision, recall, and *F*-measure values were computed as the average mean of the precision, recall, and *F*-measure values for the answered questions. Additionally, "*F*-measure global" represents the average *F*-measure over all questions. Questions that the system did not generate any SPARQL queries for were considered as unprocessed questions.

To evaluate our approach further, we investigated the impact of question type on the evaluation measures. Some queries rely on concepts that are not in the DBpedia ontology. These concepts may be in the YAGO, FOAF, or DBpedia property namespaces. Additionally, some queries may require aggregation operations (eg, ordering, counting, filtering). Therefore, we divide the questions into four categories: DBOnly-Agg (rely solely on the DBpedia ontology without any aggregation operations), DBOnly + Agg (rely solely on the DBpedia ontology with aggregation operations), NotDBOnly-Agg (rely on resources outside the DBpedia ontology without any aggregation operations), and NotDBOnly + Agg (rely on properties outside the DBpedia ontology with aggregation operations). The numbers of questions in each category and the evaluated measures for the processed questions in each category are listed in Table 3.

To evaluate the impact of the ontology lexicon on our approach, we tested two automatically generated ontology lexicons: DBlexipedia [31] and the adjective lexicalizations generated by Walter and his colleagues [32], which are denoted as AdjLex. DBlexipedia was generated using

TABLE 2 Evaluation results on the QALD-5 test dataset

	Processed	Right	Partial	Recall	Precision	*F*-measure	*F*-measure global
Ours	38	21	2	0.73	0.58	0.65	0.50
APEQ [19]	26	8	5	0.48	0.40	0.44	0.23
QAnswer [28]	37	9	4	0.35	0.46	0.40	0.30
SemGraphQA [29]	31	7	3	0.32	0.31	0.31	0.20
YodaQA [30]	33	8	2	0.25	0.28	0.26	0.18

TABLE 3 Evaluation of our approach for each question category

	Questions	Processed	Right	Partial	Recall	Precision	*F*-measure
DBOnly − Agg	25	23	15	2	0.69	0.71	0.70
DBOnly + Agg	11	8	1	1	0.16	0.63	0.26
NotDBOnly − Agg	10	6	4	0	0.67	0.67	0.67
NotDBOnly + Agg	3	1	1	0	1.00	1.00	1.00

TABLE 4 Evaluation of our approach using different ontology lexicons

Ontology lexicon	Processed	Right	Partial	Recall	Precision	*F*-measure	*F*-measure global
DBpedia ontology lexicon	38	21	2	0.73	0.58	0.65	0.49
DBlexipedia	19	4	3	0.47	0.23	0.31	0.12
DBlexipedia + Adj	19	4	3	0.47	0.23	0.31	0.12

the M-ATOLL framework [31] and contains multilingual lexicalizations for over 600 properties in the DBpedia ontology. AdjLex contains 15 380 adjective entries for 2796 properties in DBpedia 2014. It was extracted automatically using a machine learning approach. Because DBlexpdia does not include adjective lexicalizations, we added AdjLex to DBlexipedia. Table 4 lists the evaluation results for our approach when using these lexicons and the DBpedia ontology lexicon.

4.1 | Error analysis and discussion

Based on our observations, there are three key reasons for the errors associated with certain questions in our approach. First, some errors occur during the syntactical analysis step. Stanford CoreNLP generates incorrect POS tags or dependency parse trees for certain questions. For example, for all questions starting with "show me," it detects the verb "*show*" as a noun. Although we corrected a few of these known errors manually, there are other cases that were left unchanged. In our evaluations, we determined that 8% of the outputs from Stanford CoreNLP for the questions in QALD-5 were incorrect.

Second, there are some questions for which our approach cannot generate any templates. This can occur for two reasons. In some cases, a question may contain comparative, temporal, or conjunctive phrases that the current version of our approach cannot handle. These cases represent 18% of the questions. As shown in the third row of Table 3, questions that require aggregation operations have lower F-measure scores than other questions. In other cases, there is a significant structural gap between the manner in which a question is expressed and the manner in which the related information is stored in the knowledge base. For example, in the question "Who killed John Lennon?," the information required to answer the question is contained in the entity "*res:Death_of_John_Lennon*" and property "*dbp:conviction*," rather than the more probable entity "*res:John_Lennon*" and property "*dbo:killedBy*." Such structural gaps existed for 6% percent of the questions.

Finally, some errors are related to the ranking step. For 8% of the questions, the top-ranked SPARQL query generated an incorrect or partially correct answer, while another SPARQL query with a lower score generated the correct answer. This demonstrates the need for additional focus on the ranking algorithm.

In addition to the errors described above, our approach fails if the ontology lexicon does not contain an appropriate lexicalization of the concept or property that is required to answer the question. As shown in Table 4, using automatically generated ontology lexicons decreases the efficiency of the proposed approach. Our observations revealed that the incompleteness of DBlexipedia and AdjLex is the main cause of this decrease in the efficiency. Therefore, the existence of a comprehensive ontology lexicon is crucial for our approach.

5 | CONCLUSIONS

In this paper, we proposed a novel approach for QA over linked data. The proposed approach consists of six main components: syntactical analysis, template generation, named entity recognition, candidate selection, ranking, and answer retrieval. It uses an ontology lexicon to overcome the lexical gaps between phrases in questions and phrases in the knowledge base. Additionally, by using an ontology lexicon and dependency parse trees of input questions, it attempts to capture all possible mappings between an input question and the knowledge stored in the knowledge base. In most cases, more than one SPARQL query is generated for an input question. A ranking algorithm is then used to select the query with the highest score and a non-empty result. However, we still cannot answer questions

for which we do not have a correct template. Additionally, the proposed approach relies heavily on an ontology lexicon, meaning the incompleteness of an ontology lexicon decreases its performance. In the future, we will extend our templates to cover additional questions. Furthermore, we will attempt to develop a robust method for constructing ontology lexicons automatically.

ORCID

Mehdi Jabalameli https://orcid.org/0000-0002-8423-0545
Mohammadali Nematbakhsh https://orcid.org/0000-0002-4374-9228
Ahmad Zaeri https://orcid.org/0000-0003-3020-2597

REFERENCES

1. C. Unger et al., *An Introduction to Question Answering over Linked Data*, in Proc. Reason. Web. Reason. Web Big Data Era, Athens, Greece, Sept. 2014, pp. 100–140.
2. K. Höffner et al., *Survey on challenges of question answering in the semantic web*, Semant. Web J. **8** (2016), 895–920.
3. J. P. McCrae and C. Unger, *Design patterns for engineering the ontology-lexicon interface*, in Towar. Multiling. Semant. Web SE – 2, P. Buitelaar and P. Cimiano (eds), Springer, Berlin Heidelberg, 2014, pp. 15–30.
4. A. Bonifati, W. Martens, and T. Timm, *DARQL: Deep Analysis of SPARQL Queries*, in Companion Proc. Web Conf., Lyon, France, Apr. 2018, pp. 187–190.
5. G. Xiao et al., *Efficient Handling of SPARQL OPTIONAL for OBDA*, in Semant. Web – ISWC 2018, D. Vrandečić et al. (eds), Springer International Publishing, Cham, 2018, pp. 354–373.
6. P. Cimiano et al., *Towards portable natural language interfaces to knowledge bases-The case of the ORAKEL system*, Data Knowl. Eng. **65** (2008), 325–354.
7. C. Unger and P. Cimiano, *Pythia: compositional meaning construction for ontology-based question answering on the semantic web*, in Proc. Natural Language Process. Inf. Syst., Alicante, Spain, June 2011, pp. 153–160.
8. C. Unger et al., *Template-Based Question Answering over RDF Data*, in Proc. 21st Int. Conf. World Wide Web, Lyon, France, Apr. 2018, pp. 639–648.
9. A. Abujabal et al., *Automated template generation for question answering over knowledge graphs*, in Proc. Int. Conf. World Wide Web, Perth, Australia, Apr. 2017, pp. 1191–1200.
10. D. Punjani et al., *Template-Based Question Answering over Linked Geospatial Data*, in Proc. Work. Geogr. Inf. Retr. - GIR'18, Seattle, WA, USA, Nov. 2018, pp. 7:1–10.
11. J.-D. Kim and K. B. Cohen, *Natural language query processing for SPARQL generation: A prototype system for SNOMED-CT*, in Proc. BioLINK 2013 Roles Text Min. Biomed. Knowl. Discov. Transl. Med., Berlin, Germany, July 2017, pp. 32–38.
12. L. Biermann, S. Walter, and P. Cimiano, *A guided template-based question answering system over knowledge graphs*, in Proc. 21st Int. Conf. Knowl. Eng. Knowl. Manag., Nancy, France, Nov. 2018, pp. 1–4.
13. J. Berant et al., *Semantic parsing on freebase from question-answer pairs*, in Proc. Empir. Methods Nat. Lang. Process., Seattle, WA, USA, Oct. 2013, pp. 1533–1544.
14. Q. Cai and A. Yates, *Large-scale semantic parsing via schema matching and lexicon extension*, in Proc. 51st Annu. Meet. Assoc. Comput. Linguist. (Volume **1** Long Pap.), Sofia, Bulgaria, Aug. 2013, pp. 423–433.
15. M. Yahya et al., *Robust question answering over the web of linked data*, in Proc. 22nd ACM Int. Conf. Conf. Inf. Knowl. Manag. - CIKM '13, San Francisco, CA, USA, Oct. 2013, pp. 1107–1116.
16. S. He, *Question answering over linked data using first-order logic*, in Proc. Conf. Empir. Methods Nat. Lang. Process, Doha, Qatar, Oct. 2014, pp. 1092–1103.
17. S. Shekarpour et al., *SINA: Semantic interpretation of user queries for question answering on interlinked data*, J. Web Semant. **30** (2015), 39–51.
18. A. Freitas et al., *Querying linked data graphs using semantic relatedness: A vocabulary independent approach*, Data Knowl. Eng. **88** (2013), 126–141.
19. C. Unger et al., *Question answering over linked data (QALD-5)*, in Proc. Work Notes Conf. Labs Evaluation Forum, Toulouse, France, Sept, 2015, pp. 1–10.
20. C. Zhu et al., *Semantic Technology*, in A Graph Traversal Based Approach to Answer Non-Aggregation Questions over DBpedia, G. Qi (ed), Springer International Publishing, Cham, 2016, pp. 219–234.
21. T. Hamon, N. Grabar, and F. Mougin, *Querying biomedical linked data with natural language questions*, Semant. Web **8** (2017), 581–599.
22. D. Mamgai et al., *An improved automated question answering system from lecture videos*, in Proc. 2nd Int. Conf. Commun. Comput. Netw., Chandigarh, India, 2018, pp. 653–659.
23. M. Devi and M. Dua, *ADANS: an agriculture domain question answering system using ontologies*, in Int. Greater Noida, India, May, Conf. Comput. Commun. Autom., Greater Noida, India, 2017, pp. 122–127.
24. C. D. Manning et al., *The stanford CoreNLP natural language processing toolkit*, in Proc. Annu. Meet. Assoc. Comput. Linguist. Syst. Demonstr., Baltimore, MD, USA, June, 2014, pp. 55–60.
25. P. N. Mendes et al., *DBpedia spotlight : Shedding light on the web of documents*, in Proc. 7th Int. Conf. Semant. Syst., Graz, Austria, Sept. 2011, pp. 1–8.
26. G. A. Miller, *WordNet: a lexical database for English*, Commun. ACM **38** (1995), 39–41.
27. C. Unger et al., *A Lemon Lexicon for DBpedia*, in Proc. 2013th Int. Conf. NLP & DBpedia, Sydney, Australia, Oct. 2013, pp. 103–108.
28. S. Ruseti et al., *QAnswer - enhanced entity matching for question answering over linked data*, in Conf. Labs Evaluation forum, Toulouse, France, Sept. 2015.
29. R. Beaumont, B. Grau, and A. L. Ligozat, *SemGraphQA@ QALD-5: LIMSI Participation at QALD-5@CLEF*, in Conf. Labs Evaluation forum, Toulouse, France, 2015.
30. P. Baudiš and J. Šedivý, *Modeling of the question answering task in the YodaQA system* in Conf. Labs Evaluation forum, Toulouse, France, Sept, 2015, pp. 222–228.
31. S. Walter, C. Unger, and P. Cimiano, *M-ATOLL: A framework for the lexicalization of ontologies in multiple languages*, in Int. Semantic Web Conf., Riva del Garda, Italy, Oct. 2014, pp. 472–476.
32. S. Walter, C. Unger, and P. Cimiano, *Automatic acquisition of adjective lexicalizations of restriction classes: a machine learning approach*, J. Data Semant. **6** (2016), 113–123.

AUTHOR BIOGRAPHIES

Mehdi Jabalameli received his BSc and MSc degrees in Computer Software Engineering from Kharazmi University, Tehran, Iran in 2000 and Sharif University of Technology, Tehran, Iran in 2002, respectively. Currently, he is a PhD candidate with the Department of Software Engineering, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran. His research interests include question answering systems, natural language processing, and the semantic web.

Mohammadali Nematbakhsh is a Professor in the Computer Engineering Department of the University of Isfahan, Isfahan, Iran. He received his BSc degree in Electrical Engineering from Louisiana Tech University, USA, in 1981 and his MSc degree and PhD in Electrical and Computer Engineering from the University of Arizona, USA, in 1983 and 1987, respectively. He worked for the Micro Advanced Company in the USA and for the Toshiba Corporation for six years, split between the USA and Japan, before joining the University of Isfahan. He has written more than 200 published research papers and two database books that are widely used in universities. He has also registered three US patents. His main research interests include the intelligent web, database systems, and natural language processing.

Ahmad Zaeri received his BSc and MSc degrees in Computer Software Engineering from Shahid Beheshti University, Tehran, Iran in 1998 and the University of Isfahan, Isfahan, Iran in 2001, respectively. He received his PhD from the University of Isfahan, in 2012. He was a visiting researcher at the Knowledge Representation and Knowledge Management Research Group of Mannheim University, Mannheim, Germany in 2011. He was an Assistant Professor at the University of Isfahan from 2013 to 2017 before joining Wysdom.ai in Canada as a senior artificial intelligence and natural language processing architect. His main research interests include natural language processing, the semantic web, description logic, and chatbot systems.