IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 6, 2021, accepted November 15, 2021, date of publication November 25, 2021,
date of current version December 6, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3130667

Intelligent SPARQL Query Generation for Natural Language Processing Systems

YI-HUI CHEN[1,2], (Member, IEEE), ERIC JUI-LIN LU[3], (Member, IEEE), AND TING-AN OU[3]

[1]Department of Information Management, Chang Gung University, Taoyuan 33302, Taiwan
[2]Kawasaki Disease Center, Kaohsiung Chang Gung Memorial Hospital, Kaohsiung 83301, Taiwan
[3]Department of Management Information Systems, National Chung Hsing University, Taichung 402204, Taiwan

Corresponding author: Eric Jui-Lin Lu (jllu@nchu.edu.tw)

The work of Yi-Hui Chen was supported in part by the Ministry of Science and Technology of the Republic of China, Taiwan, under Grant MOST 107-2221-E-182-081-MY3 and Grant MOST 110-2221-E-182-026-MY3; and in part by the Kaohsiung Chang Gung Memorial Hospital.

ABSTRACT Developing question answering (QA) systems that process natural language is a popular research topic. Conventionally, when QA systems receive a natural language question, they choose useful words or phrases based on their parts-of-speech (POS) tags. In general, words tagged as nouns are mapped to class entities, words tagged as verbs are mapped to property entities, and words tagged as proper nouns are mapped to named entities, although the accuracy of entity type identification remains low. Afterward, the relationship between entity types as RDF types determines the first element to be a pivot word to generate the SPARQL (acronym for SPARQL protocol and RDF query language) query on the basis of the sequences by a specific graph or tree structure, such as dependence tree or directed acyclic graph (DAG). However, the generated SPARQL query is difficult to adapt to the given query request in that the sequences are decided by a fixed structure. Unlike in previous research, SPARQL generation occurs automatically according to the entity type identification and RDF type identification results. This study attempts to design a method that leverages machine learning to learn human experiences in entity type identification as well as RDF-type identification. We approach the problem as a multiclass classification problem and propose a two-stage maximum-entropy Markov model (MEMM). The first stage identifies the entity type and the second identifies the RDF type for the purpose of generating appropriate SPARQL queries to meet the query request. Along with the templates designed for the two-stage MEMM model, we develop an automatic question answering prototype system called QAWizard. The experimental results show that QAWizard outperforms all other systems in question answering when evaluated on Linked Data version 8 (QALD-8) metrics.

INDEX TERMS Question answering system (QA), parts-of-speech (POS), SPARQL query, maximum-entropy Markov model.

I. INTRODUCTION

For decades, experts from various fields have developed technologies that enable computers to understand human languages and complete tasks assigned by humans. For example, Siri from Apple Inc. and Google Assistant can understand natural language in order to help carry out tasks. Although human languages represent one of the best interfaces with computer systems, they are extremely complex. So far, computers have not yet been fully able to understand human language. Natural language question answering (QA) has thus become a popular research topic [1], [7]–[9], [23], [27].

QA systems analyze users' questions by using natural language processing (NLP) technologies and converting these questions (interrogative sentences) into query languages such as Structured Query Language (SQL) or SPARQL Protocol and RDF Query Language (SPARQL) suitable for back-end databases or knowledge bases. Because questions often include words with latent semantic meaning, and because linked data can be used to recognize latent semantics, researchers have employed linked databases (e.g., DBpedia [28], FreeBase [3], and Yago [41]) to enable computers to obtain correct answers to the questions.

A typical architecture of a natural language question answering (QA) system based on linked data (QALD; Figure1 has four components: parsing, entity mapping, SPARQL generation, and evaluation).

The associate editor coordinating the review of this manuscript and approving it for publication was Emre Koyuncu.

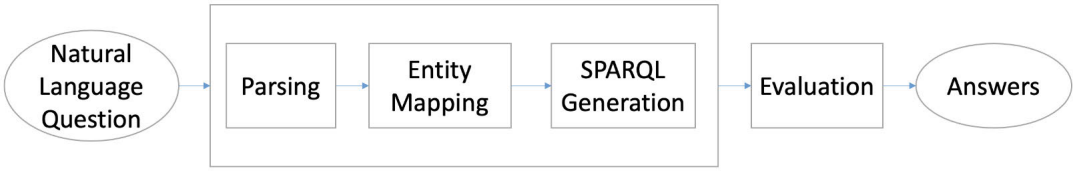


FIGURE 1. A typical architecture of QALD systems.

When parsing a question, QALD systems utilize natural language processing tools to identify words, parts-of-speech (POS) tags, lemmas, dependency trees, and named-entity recognition (NER). Next, based on those POS tags and predefined lexical rules, mapping is performed between the chosen words or phrases (denoted as w_i hereinafter) and the entities in a linked database. After entities have been identified, SPARQL queries can be generated based on the dependency (or parsing) tree structure of the question [4], [15], [16], [26]. Finally, the generated SPARQL queries are executed to retrieve answers from the linked database, which is known as evaluation.

Entities in linked databases are mostly classified into name, property, and class entities. Named entities are generally names of people, animals, events, and objects; property entities describe the relationships between named entities; and class entities describe the class of an entity. In DBpedia, each entity is represented in the form of a uniform resource identifier (URI). For example, War and Peace is linked to the URI http://dbpedia.org/resource/War_and_Peace and can be denoted as dbr:War_and_Peace. Problems such as lexical gaps and ambiguity [22] occur during entity mapping. As a result, PATTY [37], ESA [17], and Word2vec [19] were developed to alleviate these problems.

In the process of parsing, one of the most important things is to identify useful w_i from a question and assign them appropriate entity types in order to avoid excessive entity mappings. For example, if a word is correctly classified as a class type, the word does not need to be mapped to either named entities or property entities. Conventionally, the useful w_i are chosen based on their POS tags. Words tagged as nouns are mapped to class entities, words tagged as verbs are mapped to property entities, and words tagged as proper nouns are mapped to named entities [16], [21], [44]. HAWK [44] further considers either consecutive proper nouns (NNP) or two common nouns connected by a conjunction to be one named entity. Nonetheless, the accuracy of entity type identification in past research remains low.

To produce SPARQL queries based on natural language questions, studies have adopted dependency tree or directed acyclic graph (DAG) to determine the relationships between entities. For example, Freitas and Curry [15], [16] chose an NER from questions as a pivot, and transformed the dependency tree of questions based on the pivot into a sequence structure called partial ordered dependency structure (PODS), where the pivot is the first element in the sequence. Then, SPARQL queries are generated based on the sequence. However, the generated SPARQL queries have difficulty fulfilling query requests since the sequence is generated by a fixed structure.

In contrast to the aforementioned approaches, we consider parsing to be a multiclass classification problem, whether in the entity type identification process or RDF type identification process for SPARQL query generation. A popular classification algorithm within machine learning, Maximum-Entropy Markov Model (MEMM) [33], is utilized to design a two-stage MEMM model to learn human experiences in identifying the entity type as well as the RDF type. Accordingly, the SPARQL generation is automated on the basis of entity type identification and RDF type identification results. The first stage MEMM model enables us to correctly choose useful w_i from a question and determine their appropriate entity types in order to be used in the entity mapping process. Then, the second stage MEMM model is used to determine entities in the same RDF triples based on the results of the first stage classification model. Finally, predefined templates, designed for the results of the second stage model, are employed for SPARQL generation. Aside from the templates designed for the second stage MEMM model, we developed an automatic question answering prototype system called QAWizard.

This study uses the questions provided in the QALD challenges as the data source for testing and training. We use QALD-8 datasets [45] of multilingual questions for training as well as performance measurement, but exclude comparatives, superlatives, and interrogative sentences requiring URIs with skos:Category entities. Our prototype, QAwizard, achieved outstanding performances on the metrics. The accuracy of the first and second stage learning models reached 86.94% and 98.73%; respectively. On the QALD-8 dataset, the recall and F-measure of our system outperform those of other QA systems evaluated.

The rest of the paper is organized as follows. In Section 2, we describe QA-related studies. Section 3 explains the proposed approach in detail. Section 4 outlines our experiments and their results. Finally, Section 5 offers conclusions and gives suggestions for possible development in the future.

II. RELATED LITERATURE

As stated in the first section, this study attempted to design an effective approach for choosing the right w_i and assigning them appropriate entity types in the process of parsing. In addition, we employed a two-stage MEMM model, along

DBRDict-A

Label	URL
War and Peace	http://dbpedia.org/resource/War_and_Peace
Barack Obama	http://dbpedia.org/resource/Barack_Obama
United States	http://dbpedia.org/resource/United_States

DBRDict-B

Label	URL
Aena	https://dbpedia.org/resource/ENAIRE
U.S.A	http://dbpedia.org/resource/United_States
United States of American	http://dbpedia.org/resource/United_States

FIGURE 2. Example data for DBRDict-A and DBRDict-B.

with predefined templates, to generate appropriate SPARQL queries in the process of SPARQL generation. Therefore, we reviewed what had been done in these two processes in subsections II-A and II-B.

A. PARSING

Conventionally, the useful w_i are selected from a question based on their POS tags. Only proper nouns, verbs, and nouns were used. Then, Proper-Noun-tagged words are mapped to named entities, Verb-tagged words are mapped to property entities, and Noun-tagged words are mapped to either class or property entities. If two nouns are connected by conjunctions, HAWK [44] treats them as one proper noun and maps it to a named entity. By using CoreNLP [32], Freitas and Curry [15] selected w_i tagged with NER and used it as a named entity. However, for a question such as ''What is in a chocolate chip cookie?'', the above approaches failed to identify ''chocolate chip cookie'' as a named entity.

WDAqua [7]–[10] used n-gram in a question to identify all entities (including named entities, class entities, and property entities) in knowledge bases. In addition to mentioned entities, Intui3 [12] also resolved demonyms by using a list of demonyns and their associated countries. Similar to WDAqua, Lu *et al.* [31] also used n-gram in a question to identify named entities. These named entities were collected from DBpedia and saved in local tables known as DBRDict-A and DBRDict-B. These tables have two columns, namely the label of named entities and its URI, as shown in Figure 2. Both tables contain named entities. The only difference is that named entities in DBRDict-B contain a property dbo:wikiPageRedirects, but that DBRDict-A does not. For example, if a question contains ''U.S.A.'', http://dbpedia.org/resource/U.S.A is found. However, this URI contains no useful information, just a redirect URI link which is http://dbpedia.org/United_States. Because http://dbpedia.org/United_States is what we really need for the question, its URI in DBRDict-B is thus http://dbpedia.org/United_States.

Hu *et al.* [23], [48] developed a QA system called gAnswer2. gAnswer2 pre-built two large dictionaries, namely entity mention dictionary and relation mention dictionary. These dictionaries can be used to efficiently identify named entities and property entities from a question. However, the entity types are recognized by fixed rules. The fixed rules cannot adapt to all input sentences. As a result, the accuracy of entity identification for the methods above are low.

In contrast to the aforementioned approaches, Xser [46] designed a structured perceptron to select the right w_i and assign them entity types. Xser considered four types of entities: named entity, category entity, relation entity, and variable. They employed three types of features, namely lexical features, POS tags features, and NER features. Although the accuracy of the structured perceptron is unknown, we believe this approach is viable and thus adopted in our work.

B. SPARQL GENERATION

SPARQL is a query language used to search data stored in an RDF format repository. To generate SPARQL queries from questions, Intui3 [12] used natural language processing tools to split each question into chunks, and then processed chunks in a right-to-left order to generate SPARQL queries. Freitas and Curry [15], [16] converted a question into a so-called partially ordered dependency structure (PODS). The construction of PODS is based on the question's dependency tree, and nodes are merged or pruned based on pre-defined rules. Then, a NER node is selected as a pivot and the generation of the SPARQL query begins from the pivot. Xser [46] utilized predicate-argument dependency structure to represent query intention, and thus created a semantic DAG. Finally, they used pre-defined rules to convert the query intention into SPARQL queries.

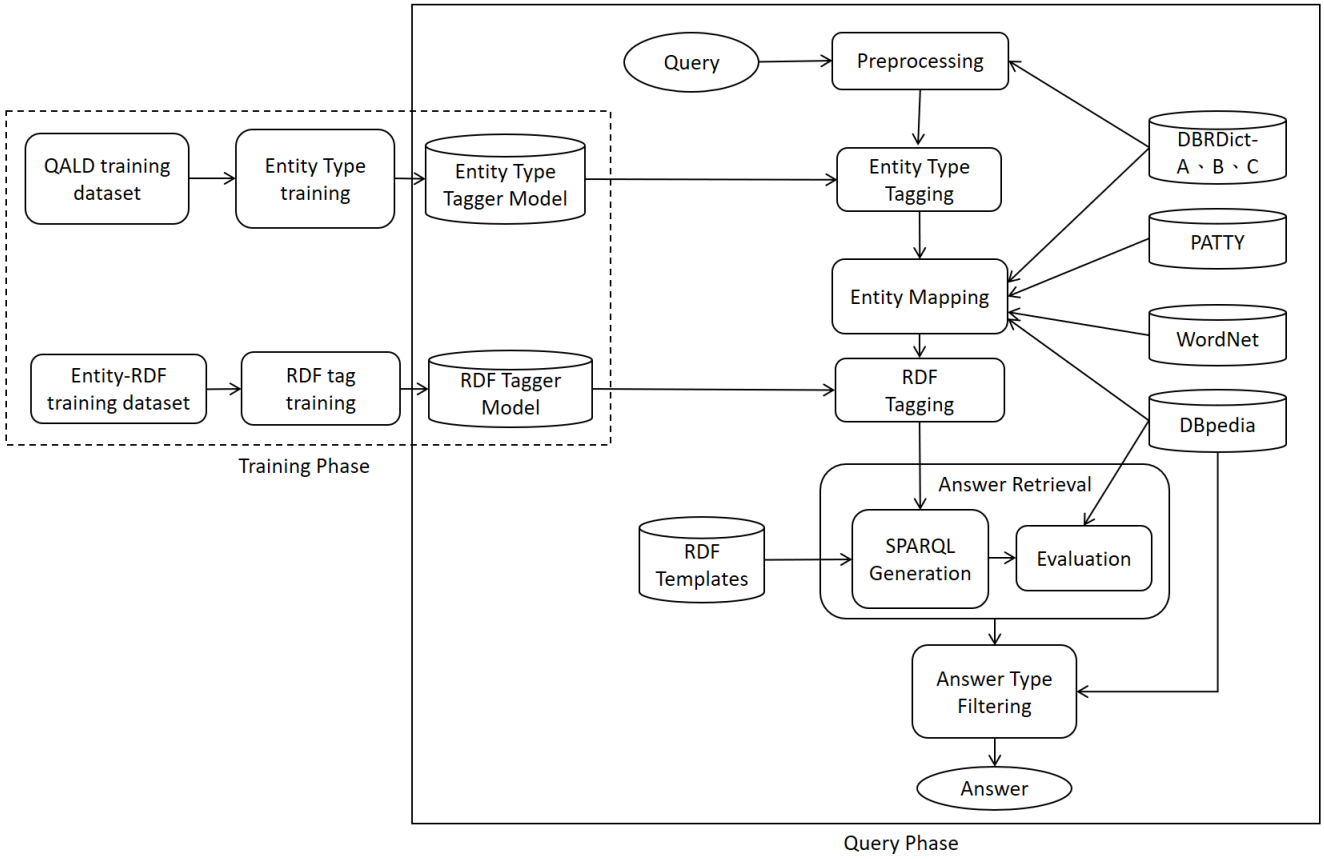


FIGURE 3. The architecture of the proposed approach.

gAnswer2 [23], [48] proposed a graph data-driven approach to generate SPARQL queries. Firstly, a so-called semantic query graph (SQG) is built based on the dependency tree structure of a question. A score is assigned to each sub-graph of SQG. Then, the top-k subgraphs are converted into SPARQL queries. After obtaining all entities in a question, WDAqua [7]–[10] compiles all possible SPARQL queries that give a non-empty result. The SPARQL query syntax is restricted to a maximum of two simple RDF triples. The idea is simple: for every pair of entities, a breath-first search of depth 2 is performed on the knowledge base. Yus *et al.* [47] adopted SPARQL-like query language to express user's requests to access local knowledge from the knowledge repositories. To efficiently explore RDF data, Gorgojo *et al.* [20] proposed a tool, named RDF Surveyor, that can deal with massive datasets. Mehmood *et al.* [34] presented an index-based approach, named QPPDs, to answer queries by exploiting linked data from multiple and distributed datasets. Liu *et al.* [30] translated the natural language query into SPARQL query to represent the query intention as a corresponding query graph. Smadi *et al.* [40] leveraged linked open data (LOD) for automatic question answering systems in the Arabic language. Using LOD, Chughtai *et al.* [5] proposed an ontology-based model to recommend reviewers for papers. Since the tree structure or query graph is constructed by a specific rule, the past researches cannot be adaptively generated based on different input sentences.

III. THE PROPOSED APPROACH

QAwizard, as shown in Figure 3, which is divided into two phases: the training phase and the query phase. In the training phase, the QALD dataset is used for training, and MEMM is employed in learning both entity type and RDF tags, including the entity type tagger model and the RDF tagger model. The entity type tagger model attempts to learn the useful w_i and their entity types from training questions. Based on the tagged results obtained in the entity type tagger model, the RDF tagger model attempts to learn which entities should be in one RDF triple and their positions (either subject, property, or object) in the triple. After training phase, the components in the query phase handle the input questions and answer the questions using process such as preprocessing, entity type tagging, entity mapping, RDF tagging, SPARQL generation, evaluation, and answer filtering.

A. TRAINING PHASE

As shown in Figure 3, there are two learning models called entity type tagger model and RDF tagger model. The input for the entity type tagger model is a manually tagged dataset based on QALD datasets. Figure 4 shows a snippet of a QALD dataset, including a question, its SPARQL query, and

```
"question": [{
        "language": "en",
        "string": "How many pages does War and Peace have?",
        "keywords": "how many, pages, War and Peace"
    }
],
"query" : {
    "sparql": "{SELECT DISTINCT ?n WHERE {dbr:War_and_Peace dbo:numberOfPages ?n .}}"
},
"answers": [{
    "results": {
        "bindings": [{
            "uri": {
                "type": "uri",
                "value": "1225"
            }
        }]
    }
}]
```

FIGURE 4. An example QALD dataset.

the corresponding answers. The question segment contains its language encoding, the question string, and the useful keywords in the question string. The query segment contains the SPARQL query that is used to look up the answers from DBpedia. The answers segment contains the answers for the question.

The main purposes of the entity type tagger model are to determine useful keywords or phrases w_i from the question sentences and assign them the right entity types. Each word or phrase is annotated as E, R, C, V, or N to represent named entity, relation entity, class entity, variable, or non-useful entity, respectively. Named entity, relation entity and class entity indicates the name of object, property/relation of object, and the class of an entity, respectively.

Take the question in Figure 4 as an example: because the keyword pages is used as dbo:numberOfPages, which is a property/relation, the entity type of pages is manually tagged as R. If w_i is a WH- words (or interrogatives), it is tagged as V. If a w_i is not matched to any entity in the SPARQL query, w_i is tagged as N. Since it is possible that one entity is represented as more than one word, we use -B (Begin), -I (Intermediate), and -E (End) to concatenate consecutive words into one entity. Noted that -E will be used only when the number of consecutive words is greater than 2. For example, the keywords "How many" are WH- words, and both words represent one entity. Thus, they are tagged as V-B and V-I; respectively. As stated in the previous section, we extracted named entities using DBRDict-A and DBRDict-B. Therefore, the keywords "War" and "Peace" are converted into War_and_Peace and tagged as one entity type, E-B. Finally, the question sentence is tagged as follows:

How/V-B many/V-I pages/R-B does/N War_and_Peace/E-B have/N

As we can see from the example, the determination of w_i (including How many, pages, War and Peace) and their entity types from a question has now become a multiclass classification problem. In this paper, we chose to use MEMM.

In an MEMM model, as shown in Equation (1), we estimate the probability that the destination state is a particular class, c (i.e. entity types or RDF types). The destination state can take on C different values corresponding to the classes c_1, $c_2, \ldots, c_C$. Here, x is a word that needs to be tagged from the features. A feature for tagging might be mapped to this word starts from its j leading words and ends with its k consecutive words (i.e., denoted by (j, k) in Table 1). For each such feature $f_{\gamma\alpha}$(see the details of feature functions defined by Equation (4) in Section III-A), we have a weight β_γ, where γ indicates the γ-th feature in the entity type tagger model and RDF type tagger model. Given the features and weights, the trained models are to choose a tag for the word, a named entity tag or an RDF tag, for example. The model chooses the tag that has the maximum probability (in Equation (3); the probability is of a particular class c given the observation x). Z, as shown in Equation (2), is a normalization factor specifying the number of features as N to make the value of the weight dependent on the class c, denoted by $\beta_{c'\gamma}$. Then, we want to find the parameters β, which maximize the likelihood of the M training samples for training our two-stage models as shown in Equation (3).

$$p(c|x) = \frac{1}{Z} exp \sum_{\gamma=1}^{N} \beta_\gamma f_{\gamma\alpha} \tag{1}$$

$$Z = \sum_{C} p(c|x) = \sum_{c' \in C} exp \sum_{\gamma=1}^{N} \beta_{c'\gamma} f_{\gamma\alpha} \tag{2}$$

$$\hat{\beta} = \arg\max_{\beta} \prod_{\varphi=1}^{M} P(c^{(\varphi)}|x^{(\varphi)}) \tag{3}$$

The features we consider include words, their entity types, and unknown words, such as word shape, suffix, and prefix. The features of trained data as training set are summarized

in Table 1. For a given word w_i, the first j leading words, w_i and k consecutive words are individually treated as a single words, denoted as words(j, k). The combination of words(j, k) is denoted as w-sequence(j, k). The entity tagged type of w-sequence(j, k) is denoted as et-sequence(j, k). The entity types of the first j leading words of w_i combined with k consecutive words are represented as et-pair(j, k).

TABLE 1. The feature set for both entity type tagger model and RDF tagger model.

item	Feature	Description
1	words(j,k)	$w_{i-j}, \ldots w_{i-1}, w_i, w_{i+1}, \ldots w_{i+k}$
2	w-sequence(j,k)	$w_{i-j} \ldots w_{i-1} w_i w_{i+1} \ldots w_{i+k}$ in one sequence
3	et-sequence(j,k)	$et_{i-j} \ldots et_{i-1} et_i et_{i+1} \ldots et_{i+k}$ in one sequence
4	et-pair(j,k)	et_{i-j} et_{i+k} in one sequence
5	word-et(0,j)	w_i et_{i+j} in one sequence
6	wordshapes(j,k)	$\text{wordshape}_{i-j} \ldots \text{wordshape}_{i-1}$ wordshape_i wordshape_{i+1} $\ldots \text{wordshape}_{i+k}$
7	Suffix/prefix	suffix(w_i) and prefix(w_i)

Among them, features word-shape, suffix and prefix are very useful for realizing the unknown words [26]. They convert lowercase letters to 'x', uppercase letters to 'X', and numbers to 'd'. For example, pages is converted to xxxxx. Also, shorter word shape features are used. That is, consecutive 'x's, 'X's, or 'd's are removed, and pages is converted to x.

We continue to use "How many pages does War and Peace have?" as an example, and the current word w_i is assumed to be pages. Based on the tagged result, w_{i-1} is many, w_{i+1} is does, et_i is R-B, et_{i-1} is V-I, and et_{i+1} is N. The features to be used in MEMM, if w_i is pages, are shown in in Table 2.

TABLE 2. An example feature set for the entity type tagger model.

item	Feature	Description
1	words(-1,1)	3 words: many, pages, does
2	w-sequence(-1,1)	word sequence: "many pages does"
3	et-sequence(-1,1)	entity type sequence: "V-I R-B N"
4	et-pair(-1,1)	its corresponding sequence: "V-I N"
5	word-et(0,1)	word and its entity type: "pages N"
6	wordshapes(-1,1)	word shapes: xxxx, xxxxx, xxxx
7	Suffix/prefix	suffix(w_i):e, ge, age prefix(w_i): p, pa, pag

An RDF triple is of the form (Subject, Predication, Object) where the subject and the predicate can only be URI, and the object can be URI or literal. In general, relation entities will be placed in the position of the property, and named entities will be placed in the position of either the subject or the object. Since SPARQL queries contain RDF triples, and since the results of the entity type tagger include named entity types, class entity types, and relation entity types, we attempted to learn which entity types are in one RDF triple. For example, in the previous example, we have tagged results as follows:

V-B V-I R-B N E-B N

Because V and N are not used in SPARQL queries, and because dbr:War_and_Peace and dbo:numberOfPages are in one RDF triple as shown in the query segment of Figure 4, we attempt to determine whether R and E are in one RDF triple through the second stage of the classification model – the RDF tagger model.

The RDF tagger model, determines the relationships between the entities. Only E, R, and C are considered; in other words, the other non-useful tags are removed after the first stage. In the RDF tagger model, we consider RDF tags such as S (Subject), O (Object), P (Property entity), and T (Class entity). For S, O, and P, we also append a number, which is used to indicate whether or not two tags are in the same RDF triple. Thus, from the previous example, the training data has now become:

E/S1 R/P1.

The number 1 indicates both S and P are in the same RDF triple. More than one RDF triple is required to obtain correct answers for complex questions. For example, after being processed by the entity type tagger and non-useful tags are removed, the tagged result of "Who are the parents of the wife of Juan Carlos I?" is R R E. Because the RDF triples (or SPARQL query) for "Who are the parents of the wife of Juan Carlos I?" are

(dbr:Juan_Carlos_*I_of_Spain*, *dbo*:*spouse*, ?*ans*)
(?ans, dbo:parent, ?uri)

we have to manually tag R R E as follows:

R/P2 R/P1 E/S1

In other words, E (dbr:Juan_Carlos_I_of_Spain) should be evaluated with the second R (dbo:spouse), not the first R (dbo:parent), because they have the same number. The RDF tag T is very special, because it is always converted into a standalone RDF triple of the form (?uri, rdf:type, T). Thus, there is no need to append a number to T. For example, after processing by the entity type tagger and removing non-useful tags, the tagged result of "Which movies did Kurosawa direct?" is C E R. Since the RDF triples for "Which movies did Kurosawa direct?" are:

(?uri, rdf:type, dbo:Film)
(?uri, dbo:director, dbr:Akira_Kurosawa)

we have manually tag C E R as follows:

C/T E/O1 R/P1

As we can see from the examples, the RDF model is now a multiclass classification model, which is totally different from other graph-based or tree-based approaches proposed so far. The feature set for the RDF tagger model is identical to the entity type tagger model. We continue to use "How many pages does War and Peace have?" as an example; the input training data for RDF tagger model is now E/S1 R/P1. If the current word w_i is assumed to be R, the features to be used in MEMM are summarized in Table 3.

TABLE 3. An example feature set for the RDF tagger model.

item	Feature	Description
1	words(-1,1)	2 words: R, E
2	w-sequence(-1,1)	word sequence: "R E"
3	et-sequence(-1,1)	entity type sequence: "P1 S1"
4	et-pair(-1,1)	the corresponding sequence: "S1"
5	word-et(0,1)	RDF tag sequence: "R S1"
6	wordshapes(-1,1)	2 word shapes: X, X
7	Suffix/prefix	suffix(w_i): R prefix(w_i): R

TABLE 4. A pre-processing example.

Token	POS tagger	Lemmatization
Where	WRB	Where
in	IN	in
France	NNP	France
is	VBZ	is
sparkling	JJ	sparkling
wine	NN	wine
produced	VBN	produce

The features listed in Table 1 are also the features of the RDF tagger model. The features defined by Table 1 are for tagging RDF types, where the outcome results of the first stage are the input data for the second stage. Thus, the output of the entity type tagger model is the input of the RDF tagger ones. For example, as shown in Table 3, the w-sequence(−1, 1) is the entity type outputted by the first stage (i.e. R E) and et-sequence(−1, 1) is the RDF type of w-sequence(−1, 1). Take "How many pages does War and Peace have?" as an example: the input training data for the RDF tagger model is E/S1 R/P1. If the current word w_i is assumed to be R, the features in Table 2 are tagged with RDF tags as the features used in MEMM and summarized in Table 3. As in the previous statement, the question is tagged as R/P1 E/S1. In Item 1 of Table 3, there is no w_{-1}, and the individual entity types of w_0 and w_1 are R and E so that words(−1, 1) is described as R, E. Item 2 of Table 3 concatenates the results of Item 1; thus, w-sequence(−1, 1) is described as R E. Item 3 (feature et-pair(−1, 1)) of Table 3 shows the words w_{-1} and w_1 of RDF types as " " S1. The word-et(0,1) feature indicates the word w_0 and the RDF type of word w_1, denoted as R S1. Meanwhile, wordshapes(−1, 1) refers to the word shapes of w_{-1}, w_0 and w_1 as X, X (since w_{-1} is " "). The Suffix/prefix feature depicts the current word as R.

The features listed in Table 1 can be defined by Equation (4), where γ is the γ-th item feature in Table 1 and Table 3, $\alpha = <b, s>$, and $f_{\gamma\alpha()}$ is the feature function. That is, the features are expressed in pairs $<b, s>$, where b is the feature of observation as the γ-th item in Table 1 and s is the destination state. Take the third item feature in Table 1: the feature function is shown as $f_{3<\text{et-sequence(-1,1), VI R-B N}>}$ ("many pages does", VI R-B N) = 1. That is, the et-sequence(−1, 1) defined in Table 2 obtains the entity types taggers of words retrieved by sentence o_t "many pages does" as VI R-B N, identical to s. Thus, the result is 1.

$$f_{i\alpha}(o_t, s_t) = f_{i<b,s>}(o_t, s_t) \tag{4}$$

$$f_{i<b,s>}(o_t, s_t) = \begin{cases} 1, & \text{if } b(o_t) = s_t \text{ and } s = s_t \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

B. QUERY PHASE

The query phase, as shown in Figure 3, presents the process for parsing the input question, assigning tags using the trained entity type tagger model, finding named entity, adding RDF tags using the trained RDF tagger model, generating SPARQL queries, and finally retrieving the answers, as described in pre-processing, entity type tagging, entity mapping, RDF tagging, SPARQL generation, evaluation and answer type filtering, respectively.

1) PRE-PROCESSING

After both the entity type tagger model and RDF tagger model have been trained, we can start processing question sentences. Pre-processed of an input question sentence includes tokenization, Part-of-Speech (POS) tagging, and lemmatization. After tokenization, the sentence can be divided into several tokens. Each token is lemmatized and then tagged with a POS tag. For example, the results after pre-processing is performed on the sentence "Where in France is sparkling wine produced?" is shown in Table 4.

To efficiently extract named entities from the sentence, we continue to use the pre-defined database DBRDict-A and DBRDict-B [31] as shown in Figure 2, which it contains label and its corresponding URL to present named entities explored by URL from DBpedia. If more than one token is matched to either DBRDict-A or DBRDict-B, they are concatenated with the notation _. For example, because the tokens "sparkling" and "wine" in Table 4 can be found in DBRDict-A, they are concatenated into one: sparkling_wine. Therefore, after pre-processing, we have:

Where in France is sparkling_wine produced

Next, we use the entity type tagger model obtained in the training phase. As a result, the sentence is tagged as:

Where/V-B in/N France/E-B is/N sparkling_wine/E-B produced/R-B.

As stated previously, the main purposes of the entity type tagger model are to select import w_i from the sentence and determine their entity types. Once both have been determined, they can be used to map appropriate entities in the DBpedia. Since only named entities (E), class entities (C), and relation entities (R) are considered, the process of entity mapping, as shown in Figure 3, is described below.

2) ENTITY MAPPING

Before going into detail, some notations should be defined. For every pre-processed sentence, important w_i and their entity types, called tokens, will be used to look up appropriate

entities in the DBpedia. The set of tokens is denoted as TK, and $tk_i \in TK$, for $1 \leq i \leq s$, where s is the size of TK. For the aforementioned example, TK contains three tokens: tk_1 is France/E, tk_2 is sparkling_wine/E, and tk_3 is produced/R. In other words, each token contains w_i and its entity type concatenated with a slash. Next, based on different types of tk_i, the tokens are applied in the lookup for appropriate entity mapping in the DBpedia as demonstrated in Algorithm 1. If the entity type is marked E, we match tk_i with named entities, and if the type is tagged with C, it is matched to the category entities. Otherwise, the type is marked R, and matched to attribute entities.

Algorithm 1 Entity Mapping Algorithm

```
procedure EntityMapping(tk_i)
    if tk_i is tagged as E then
        if tk_i ≠ labels in DBRDict-A or DBRDict-B then
            if tk_i has dbo:wikiPageDisambiguates then
                Let ∀o_i be the named entities.
    else if tk_i is tagged as C then
        if tk_i ∈ labels in DBRDict-C then
            Let matched Labels be the named entities
        else if tk_i' ∈ labels in DBRDict-C then
            Let matched labels be the named entities
    else if tk_i is tagged as R then
        if tk_i ∈ labels in PATTY then
            Let the matched labels be the name entities
```

For named entities, because both DBRDict-A and DBRDict-B contain their URIs and shown in Figure 2, it is not necessary to query DBpedia again. However, on occasion, new named entities are tagged by the entity type tagger model. For these named entities, the following steps are required: tk_i is uesed to look up entities in DBpedia, and the entities found (their rdfs:label values contain tk_i) will be checked to see if they contain a dbo:wikiPageDisambiguates property. If they do, all object values (ie. URIs) of dbo:wikiPageDisambiguates, a object's URI is denoted by o_i, will be used as named entities, instead of the original named entity. For example, if tk_i is Mary, the result of entity mapping is http://dbpedia.org/resource/Mary. Because http://dbpedia.org/resource/Mary contains a property dbo:wikiPageDisambiguates, and the property contains more than 30 object values (URIs) including (to name a few) dbr:Mary_Queen_of_Scots and dbr:Mary_II_of_England, all of the URIs are used as the entity mapping results of Mary, but http://dbpedia.org/resource/Mary itself is not used.

To reduce query overhead for DBpedia, we also created a local database called DBRDict-C. DBRDict-C is similar to DBRDict-A, but contains all class entities defined in DBpedia. Similar to named entities, tk_i of type C are mapped to the "Label" column of DBRDict-C via (sbu)string matching. If not found, both synonyms and derivatively related forms of tk_i, denoted by tk_i', are mapped to the DBRDict-C again. For example, if tk_i is car, nothing can be found in the DBRDict-C. However, the synonyms of car in WordNet include auto, automobile, machine, and motorcar; and automobile can be mapped to dbo:Automobile in the DBRDict-C.

The tk_i of type R often cannot be directly mapped to a DBpedia entity due to lexical gaps [22]. Typically, a dataset such as PATTY [37] or gAnswer2 [23] is used to resolve this type of problem. In essence, PATTY searched for important co-occurrences (or relatedness) of verbs (the "Pattern" column of Table 5) and relation entities (the "Relation" column in Table 5) from DBpedia, and generated datasets to resolve lexical gaps. A sample snippet of data extracted from PATTY is shown in Table 5. For the same example, the tk_i produced by type R is looked up in the dataset, and dbp:manufacturer, dbo:musicComposer, as well as dbo:wineProduce can be obtained.

TABLE 5. Example segments of relations and their corresponding patterns.

pattern	relation
Produced	dbp:manufacturer
Produced	dbo:musicComposer
Produced	dbo:wineProduce

3) RDF TAGGING

To generate appropriate SPARQL queries, we not only need the entity URIs from the entity mapping process, but also the tagged results from the RDF tagger model. After removing V and N, the output of the entity type tagger will be processed by the RDF tagger. For the example in which France/E, sparkling_wine/E, and produced/R, the tagged result of E E R is E/S2 E/S1 R/P1. After removing the digit numbers, we have two groups. The first group is S, and the second group is S P.

4) SPARQL GNERATION

Based on all possible tagged results, we designed 7 types of RDF triple templates, which are summarized in Table 6. The design rules for the templates are quite straightforward: Each group of an RDF tagged result will have two RDF triples because S or O can be placed in either subject or object positions. The only exception is T, which is a class entity. Because many evaluated QALD systems (if not all) can only process simple questions, it is assumed that only simple questions will be processed in our system. Thus, only the subject or the object of an RDF triple can be an answer. As a result, we can put a variable ?ans in either subject or object position. Because T can only be placed in the object position, (?ans, rdf:type, T) was also designed.

Because S, O, P, and T all have known URIs from the entity mapping processing, they will be replaced with URIs in RDF triple templates. Variables are used in the positions without known entities. For ease of reading, we use variable names such as ?P, ?S, and ?O to represent property, subject, and object; respectively.

For example, by looking at Table 6 above, the first group S is mapped to the template number 4, and thus two RDF

TABLE 6. The RDF triple templates.

Item	RDF tags	RDF triple templates
1	S P	(S, P, ?ans), (?ans, P, S)
2	P O	(O, P, ?ans), (?ans, P, O)
3	S O O S	(S, ?P, O) (O, ?P, S)
4	S	(?ans, ?P, S), (S, ?P, ?ans)
5	O	(?ans, ?P, O), (O, ?P, ?ans)
6	P	(?ans, P, ?O), (?S, P, ?ans)
7	T	(?ans, rdf:type, T)

triples – (?ans, ?p, S) and (S, ?p, ?ans) – are created. Then, the second group S P is mapped to the template item number 1, and two RDF triples – (S, P, ?ans) and (?ans, P, S) – are created. Finally, each RDF triple in one group will be concatenated into one RDF triple in another group. As a result, four SPARQL queries are created which are shown in Table 7.

TABLE 7. Four SPARQL query templates.

Number	Templates of SPARQL queries
QT1	(S,P,?ans), (?ans,?p,S)
QT2	(S,P,?ans), (S,?p,?ans)
QT3	(?ans,P,S), (?ans,?p,S)
QT4	(?ans,P,S), (S,?p,?ans)

Based on the results of entity mapping, it is assumed that France and sparkling_wine are mapped to dbr:France and dbr:Sparkling_wine, respectively. Additionally, produced is mapped to dbo:musicComposer and dbo:wineProduce. Noted that there are more than two relation entities in practice. By plugging these entity URIs into four SPARQL query templates shown in Table 7, we have eight SPARQL queries, which are listed in Table 8. For example, since dbr:France was tagged as S2, it would be plugged into (?ans, ?p, S) and become (?ans, ?p, dbr:France) in QT1. Similarly, dbr:Sparkling_wine and dbo:musicComposer were tagged as S1 and P1, respectively. They would be plugged into (S,P,?ans) and become (dbr:Sparkling_wine, dbo:musicComposer, ?ans). The SPARQL query generated for this template is shown in QR1 of Table 8, including four fields, template number (T_no), SPARQL query number (SQ_no), SPARQL queries (SQ), and query results (QR). After all SPARQL queries are generated, they will be used to query the remote DBpedia knowledge base. The query results are shown in the last column (QR) of Table 8. As you can see, there only one result is obtained.

5) EVALUATION AND ANSWER TYPE FILTERING

If there is more than one query result, it is necessary to check whether the result is correct. We used w_i which is tagged as V to determine if a result is of the right rdf:type. If w_i is "What", "Which", "Give", "List", or "Show", it is not necessary to check result types. If w_i is "Who", the appropriate rdf:type in DBpedia is either dbo:Person or dbo:Organisation. If w_i is "Where", the appropriate rdf:type is dbo:Place. If w_i is "When", the appropriate rdf:type is either xsd:date or xsd:dateTime. If w_i is of type auxiliary verb (ex. do/does, be, etc.), the result is either True or False. In our system, if the query result is not empty, "True" is given; otherwise, "False" is given. If the first word of w_i is "How", the query result is generally a value. Thus, we first check to see whether the query result is a literal of numeric type such as xsd:integer. If none exists, we count the number of results to obtain a number. For example, if a book entity has the property dbo:numberOfPages and its value is of type xsd:integer, then the value then is our answer. If a person entity has the property dbo:child and its value is not a number, we then count the number of objects as our answer.

TABLE 8. Example SPARQL queries and their result.

T_no	SQ_no	SQ	QR
QT1	QR1	(?ans ,?p, dbr:France) (dbr:Sparkling_wine , dbo:musicComposer, ?ans)	no result
	QR2	(?ans, ?p, dbr:France) (dbr:Sparkling_wine, dbo:wineProduced, ?ans)	no result
QT2	QR3	(dbr:France, ?p, ?ans) (dbr:Sparkling_wine, dbo:musicComposer, ?ans)	no result
	QR4	(dbr:France, ?p, ?ans) (dbr:Sparkling_wine, dbo:wineProduced, ?ans)	no result
QT3	QR5	(?ans, ?p, dbr:France) (?ans,dbo:musicComposer, dbr:Sparkling_wine)	no result
	QR6	(?ans, ?p, dbr:France) (?ans,dbo:wineProduced, dbr:Sparkling_wine)	dbr:Loire_Valley_(wine)
QT4	QR7	(dbr:France, ?p, ?ans) (?ans,dbo:musicComposer, dbr:Sparkling_wine)	no result
	QR8	(dbr:France, ?p, ?ans) (?ans,dbo:wineProduced, dbr:Sparkling_wine)	no result

If the query is "Where in France is sparkling wine produced?", "Where" was tagged as V by the entity type tagger, and thus the type of query result should be of type dbo:Place. Because the query result of QR6 in Table 8 is dbr:Loire_Valley_(wine), and dbr:Loire_Valley_(wine) is of type dbo:Place, it is our final answer.

IV. EXPERIMENTAL RESULTS

Java was employed to develop the system used in this study. During preprocessing, Stanford CoreNLP 3.9.1 was used in preprocessing for word segmentation, POS tagging, and lemmatization. The Part-of-Speech Tagger 3.9.1 developed by Stanford was used for MEMM training.

A. DATA

The multilingual question datasets from both QALD-7 and QALD-8 were used as the experiment datasets. The QALD-7 dataset comprises 241 sentences in the training set and

43 sentences in the test set. These 43 sentences contain comparatives, superlatives, and skos:Category entities. Because this study did not consider the aforementioned sentence types, those types of sentences were excluded, leaving 23 questions in the test set. Similarly, the QALD-8 dataset has 219 sentences in the training set and 41 sentences in the test set. Only 33 of 41 sentences were used in the experiments. We conducted the query experiment between June 28 and August 10, 2019 utilizing the official online DBpedia service.

Additional datasets employed including gAnswer2, DBRDict-A, DBRDict-B, and DBRDict-C. The so-called gAnswer2 dataset generated by Hu *et al.* [23] is basically an up-to-date version of PATTY based on DBpedia 2016. The tables DBRDict-A, DBRDict-B, and DBRDict-C were created based on DBpedia 2016, and they contain 4,872,244 named entities, 6,883,195 entities with URL redirection, and 760 class entities, respectively.

B. EXPERIMENTS

In Experiment-1, we calculated the accuracy of the trained entity-type tagger model. In Experiment-2, we calculated the accuracy of the trained RDF tagger model. In Experiment-3, we fully tested our system and compared the performance of our system with other QALD systems using the same test set.

1) EXPERIMENT-1: ACCURACY OF THE ENTITY-TYPE TAGGER MODEL

As stated previously, we designed many features for the entity-type tagger model. Because our training dataset was small (a couple of hundred sentences), unknown word features, such as word-shapes, prefix, and suffix, were used. Because the feature functions for natural language processing have been well developed, we used L3W feature set, which is similar to left-3-words as suggested by the Stanford development team. In addition to L3W, we also used L5W, B3W, and B5W. All four different feature sets are listed in Table 9.

TABLE 9. Four different feature sets.

Name of feature set	Features
L3W	words(-1,1), et-sequence(2) , wordshapes(-1,1), suffix/prefix
L5W	words(-2,2), et-sequence(2) , wordshapes(-2,2), suffix/prefix
B3W	words(-1,1), et-sequence(-2,2), et-pair(-1,1),word-et(0,-1),word-et(0,1), w-sequence(-1,1),wordshape(-1,1), suffix/prefix
B5W	words(-2,2), et-sequence(-2,2), et-pair(-1,1),word-et(0,-1), word-et(0,1), w-sequence(-1,1), wordshapes(-2,2), suffix/prefix

The accuracy of entity-type tagger model is defined as follows:

$$\text{accuracy} = \frac{\text{the number of words labeled with correct tags}}{\text{the number of words in a sentence}} \quad (6)$$

For example, if the question "How many pages does War and Peace have?" was tagged as How/V-B many/V-I pages/C-B does/N War_and_Peace/E-B have/C-B, "pages" was incorrectly tagged as C which it should have been tagged as R based on the answer provided by QALD. Similarly, "have" was tagged as C, yet "have" is not used in SPARQL query as shown in Figure 4, suggesting that its correct entity-type tag should be N. To calculate the accuracy of entity type tagging for this example, the number of words with correct tags is 6 and the total number of words in this example is 8. Therefore, the accuracy for this example is 6/8.

The experimental results for QALD-8 using L3W, L5W, B3W, and B5W are summarized in Table 10, including three fields, dataset, name of feature set (NFS), and average accuracy of entity-type tagging (Avg_accuracy). It can be easily seen that B5W has the highest accuracy on both the QALD-7 (89.86%) and QALD-8 (86.94%) datasets. Accordingly, we conducted the subsequent experiments using the entity-type tagger model trained using B5W.

TABLE 10. Accuracy of the entity-type tagger model using different feature sets.

Dataset	NFS	Avg_accuracy
QALD-7	L3W	84.78%
	L5W	86.23%
	B3W	86.23%
	B5W	89.86%
QALD-8	L3W	85.71%
	L5W	86.94%
	B3W	86.94%
	B5W	86.94%

2) EXPERIMENT-2: ACCURACY OF THE RDF TAGGER MODEL

The feature sets used in Experiment-1 are also employed in this experiment. The accuracy of the RDF tagger model was defined as follows:

$$accuracy_{RDF} = \frac{\text{the number of correct RDF tags}}{\text{the number of RDF tags in a sentence}} \quad (7)$$

Accuracy was calculated by comparing the tagging results with the standard answer in the datasets. The correctness of RDF tags was determined as follows:

a) When the entity-type tagging results in R, C, or E, the correct corresponding RDF tag should be P, T, or S/O; respectively.
b) When any two entities are in one RDF triple, their RDF tags should have the same digit number.

For example, if the question "How many pages does War and Peace have?" was tagged as V-B V-I R-B N E-B N by the entity-type tagger and subsequently tagged as P1 S1 by the RDF tagger, the accuracy would be 1 because R and E were tagged as P and S/O, and P and S has the same digit number of 1. Noted that if the RDF tagging result were P2 S2 for the question, the accuracy would still be 1 because they still have the same digit number. However, if the RDF tagging results in T S1, the R tag would have been incorrectly tagged as T, and the accuracy would be 1/2.

As shown in Table 11, regardless of which feature set was used, the RDF tagger model has the same performance on both QALD-7 and QALD-8. For consistency, the RDF tagger model trained using B5W (identical to the entity type model) was used in subsequent experiments.

TABLE 11. Accuracy of the RDF tagger model using different feature functions.

Dataset	NFS	Avg_accuracy
QALD-7	L3W	97.92%
	L5W	97.92%
	B3W	97.92%
	B5W	97.92%
QALD-8	L3W	98.73%
	L5W	98.73%
	B3W	98.73%
	B5W	98.73%

3) EXPERIMENT-3: THE OVERALL PERFORMANCE OF THE PROPOSED SYSTEM

QAWizard utilized a two-stage MEMM model to resolve the problems of entity type identification and SPARQL query generation as a multiclass classification problem. In Experiment 3, we also assessed the overall performance of our system using the QALD-8 test multilingual question set. As done in Usbeck *et al.* [45], QAKIS, gAnswer2, and WDAqua were evaluated in QALD-8, with both gAnswer and WDAqua winning the QALD-8 challenge.

The performance of a QALD system can be measured based on precision, recall, and F1-measure as proposed by Usbeck *et al.* [45]. The recall and precision were defined as follows, where gold standard answers refer to the correct answers of the questions in the QALD test set:

$$Recall(q) = \frac{\text{number of correct system answers for } q}{\text{number of gold standard answers for } q} \tag{8}$$

$$precision(q) = \frac{\text{number of correct system answers for } q}{\text{number of system answers for } q} \tag{9}$$

We also assess the overall performance of our system using the QALD-8 test multilingual question set. As found in Usbeck *et al.* [45], QAKIS, gAnswer2 and WDAqua were evaluated in QALD-8, and both gAnswer and WDAqua won the QALD-8 challenge. The system QAKIS, gAnswer, and WDAqua were all design for answering simple question without comparatives, superlatives, and interrogative sentences requiring URIs with sko:Category entities.

Table 12 shows the average of precision (Avg_P), average of recall (Avg_R), and average of the F1-measure (Avg_F1).

TABLE 12. Comparison of the proposed system (QAWizard) with other QA systems using the QALD-8 test set of multilingual questions.

System	Avg_P	Avg_R	Avg_F1
QAWizard	0.315	0.633	0.421
QAKIS	0.061	0.0528	0.0566
gAnswer2	0.3862	0.3902	0.3882
WDAqua	0.3912	0.4065	0.3987

The performances of QAWizard's Avg_R and Avg_F1 were 63.3% and 42.1%, respectively, indicating that the proposed QAWizard outperformed QAKIS, gAnswer2, and WDAqua.

However, the Avg_P of our system is slightly lower than those of gAnswer2 and WDAqua. After further analysis, we observed that the process of entity mapping due to the limitations of PATTY design. For example, in a QALD-8 sentence "How much is the total population of the European Union?", although our system correctly tagged "total population" as a property P, it failed to find the correct property entity "dbp:populationTotal" due to the limitation of PATTY (neither "total" nor "population" is a verb). It is likely that both gAnswer2 and WDAqua have built dictionaries that considered both verb and noun words.

In summary, the impacts of the proposed work are as follow:

1) Unlike the past researches, the entity type identification and RDF tagger identification in this work are dynamic processes trained by the QALD-7 and QALD-8 datasets. Thus, the entity type and RDF tagger are adaptive according to the experiences of the training data. The average accuracy of entity type identification on QALD-7 and QALD-8 are high; 86.77% and 86.63%, respectively. The accuracy of RDF tagger identification remains high at 98.73%. The positive outcomes in the experimental results confirm the feasibility of our approach.
2) Instead of utilizing the dependency tree structure of questions, SPARQL generation is automated to pre-define the general syntax generated on the basis of entity type identification and RDF type identification results.

V. CONCLUSION

The objective of a QA system is to understand a question entered by a user and then answer that question. However, many challenges exist in developing a robust QALD system [22]. In past researches, the accuracy of entity type identification has remained low. Moreover, prior SPARQL query generation techniques have been based on sequences produced by a fixed rule, such as a dependency tree or DAG, such that they cannot adapt to all question statements. To resolve such issues, this paper studies how to identify correct words or phrases from natural language questions, assigning appropriate entity types to them, and then determine the relationships between entities.

We proposed a two-stage classification model that includes entity-type and RDF tagging models. The entity type tagging model enables the designed system to select the correct words/phrases from natural language questions and assign the correct entity types to them. The RDF tagging model enables the system to determine the relationships between entities identified in the first stage, so that they can be used to generate SPARQL queries based on the designed templates.

The experimental results demonstrate that the tagger models used in the first and second stages improve accuracy

rates to 89.86% and 98.73%, respectively. Based on the RDF tagging model and the designed templates, the F1-measures for QALD-8 of the proposed system is 0.421. The outstanding performance surpasses that of all other QALD-8 systems evaluated.

Although the performance of the proposed system is strong, there is room for further improvement. For instance, the architecture of the proposed system needs to be modified so that it can work with the evaluation tool GERBILL QA platform [44]. Furthermore, the execution time of the proposed system might last up to a few hours because too many entities are chosen during the process of entity mapping; therefore, a pruning mechanism is required. Lastly, like most QALD systems developed so far, the proposed system cannot process complex questions, such as comparatives and superlatives questions. Further research should be conducted to overcome the aforementioned limitations.

REFERENCES

[1] A. Ait-Mlouk and L. Jiang, "KBot: A knowledge graph based ChatBot for natural language understanding over linked data," *IEEE Access*, vol. 8, pp. 149220–149230, 2020.

[2] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, "DBpedia—A crystallization point for the web of data," *Web Semantics*, vol. 7, no. 3, pp. 154–165, 2009.

[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2008, pp. 1247–1250.

[4] Y. Chen, M. M. Kokar, and J. J. Moskal, "SPARQL query generator (SQG)," *J. Data Semantics*, vol. 10, nos. 3–4, pp. 291–307, Dec. 2021.

[5] G. R. Chughtai, J. Lee, M. Shahzadi, A. Kabir, and M. A. S. Hassan, "An efficient ontology-based topic-specific article recommendation model for best-fit reviewers," *Scientometrics*, vol. 122, no. 1, pp. 249–265, Jan. 2020.

[6] M. Collins, "Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms," in *Proc. ACL-Conf. Empirical Methods Natural Lang. Process.-Assoc. Comput. Linguistics*, vol. 10, 2002, pp. 1–8.

[7] D. Diefenbach, A. Both, K. Singh, and P. Maret, "Towards a question answering system over the semantic web," *Semantic Web*, vol. 11, no. 3, pp. 421–439, Feb. 2019.

[8] D. Diefenbach, V. Lopez, K. Singh, and P. Maret, "Core techniques of question answering systems over knowledge bases: A survey," *Knowl. Inf. yst.*, vol. 55, no. 3, pp. 529–569, 2017.

[9] D. Diefenbach, K. Singh, and P. Maret, *WDAqua-Core0: A Question Answering Component for the Research Community, Semantic Web Evaluation Challenge*. Springer, 2017, pp. 84–89.

[10] D. Diefenbach, K. Singh, and P. Maret, "WDAqua-core1: A question answering service for RDF knowledge bases," in *Proc. 1st Int. Workshop Hybrid Question Answering With Structured Unstructured Knowl. (HQA)*, Lyon, France, Apr. 2018, pp. 1087–1091.

[11] C. Dima, "Intui2: A prototype system for question answering over linked data," in *Proc. CLEF Working Notes*, 2013, pp. 1–12.

[12] C. Dima, "Answering natural language questions with intui3," in *Proc. CLEF Working Notes*, 2014, pp. 1201–1211.

[13] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, and J. Sachs, "Swoogle: A search and metadata engine for the semantic web," in *Proc. 13th ACM Int. Conf. Inf. Knowl. Manage.*, Washington, DC, USA, Dec. 2004, pp. 652–659.

[14] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, and N. Schlaefer, "Building Watson: An overview of the DeepQA project," *AI Mag.*, vol. 31, no. 3, pp. 59–79, 2010.

[15] A. Freitas and E. Curry, "Natural language queries over heterogeneous linked data graphs: A distributional-compositional semantics approach," in *Proc. 19th Int. Conf. Intell. User Interface*, Feb. 2014, pp. 279–288.

[16] A. Freitas, J. G. Oliveira, S. O'Riain, J. C. P. da Silva, and E. Curry, "Querying linked data graphs using semantic relatedness: A vocabulary independent approach," *Data Knowl. Eng.*, vol. 88, pp. 126–141, Nov. 2013.

[17] E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using Wikipedia-based explicit semantic analysis," in *Proc. Int. Joint Conferences Artif. Intell. (IJCAI)*, Catalonia, Spain, Jan. 2011, pp. 1606–1611.

[18] D. Gerber and A. C. N. Ngomo, "Extracting multilingual natural-language patterns for RDF predicates," in *Proc. Int. Conf. Knowl. Eng. Knowl. Manage.*, Berlin, Germany: Springer, Oct. 2012, pp. 87–96.

[19] Y. Goldberg and O. Levy, "word2vec explained: Deriving Mikolov et al.'s negative-sampling word-embedding method," Feb. 2014, *arXiv:1402.3722*.

[20] G. Vega-Gorgojo, L. Slaughter, B. M. Von Zernichow, N. Nikolov, and D. Roman, "Linked data exploration with RDF surveyor," *IEEE Access*, vol. 7, pp. 172199–172213, 2019.

[21] S. He, S. Liu, and J. Zhao, "CASIA@ QALD-3: A question answering system over linked data," in *Proc. CLEF Working Notes*, 2013, pp. 1–9.

[22] K. Höffner, S. Walter, E. Marx, R. Usbeck, J. Lehmann, and A. Ngonga, "Survey on challenges of question answering in the semantic web," *Semantic Web J.*, vol. 8, no. 6, pp. 1–26, 2017.

[23] S. Hu, L. Zou, J. X. Yu, H. Wang, and D. Zhao, "Answering natural language questions by subgraph matching over knowledge graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 824–837, May 2018.

[24] Z. Hu, J. Luo, C. Zhang, and W. Li, "A natural language process-based framework for automatic association word extraction," *IEEE Access*, vol. 8, pp. 1986–1997, 2020.

[25] S. Jaf and C. Calder, "Deep learning for natural language parsing," *IEEE Access*, vol. 7, pp. 131363–131373, 2019.

[26] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2009.

[27] H. Y. Kuo, "A natural language querying system based on semantic parsing," M.S. thesis, National Chung Hsing Univ., Taichung, Taiwan, 2017.

[28] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "DBpedia—A large-scale, multilingual knowledge base extracted from Wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[29] H. Li, Y. Wang, G. de Melo, C. Tu, and B. Chen, "Multimodal question answering over structured data with ambiguous entities," in *Proc. 26th Int. Conf. World Wide Web Companion (WWW) Companion*, 2017, pp. 79–88.

[30] J. Liu, W. Li, L. Luo, J. Zhou, X. Han, and J. Shi, "Linked open data query based on natural language," *Chin. J. Electron.*, vol. 26, no. 2, pp. 230–235, Mar. 2017.

[31] E. J. Lu, H. Y. Kuo, and T.-A. Ou, "A name-entity linker for question answering over linked data," in *Proc. TANET*, 2017, pp. 986–990.

[32] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkle, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics, Syst. Demonstrations, ACL (Syst. Demonstrations)*, 2014, pp. 55–60.

[33] D. Jurafsky and J. H. Martin, *Maximum-Entropy Markov Model, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. New York, NY, USA: Draft, Jul. 2007.

[34] Q. Mehmood, M. Saleem, R. Sahay, A.-C.-N. Ngomo, and M. D'Aquin, "QPPDs: Querying property paths over distributed RDF datasets," *IEEE Access*, vol. 7, pp. 101031–101045, 2019.

[35] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, "DBpedia spotlight: Shedding light on the web of documents," in *Proc. 7th Int. Conf. Semantic Syst. I-Semantics*, 2011, pp. 1–8.

[36] G. Miller, *WordNet: An Electronic Lexical Database*. Cambridge, MA, USA: MIT Press, 1998.

[37] N. Nakashole, G. Weikum, and F. Suchanek, "PATTY: A taxonomy of relational patterns with semantic types," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn.*, Jeju Island, South Korea: Association for Computational Linguistics, Jul. 2012, pp. 1135–1145.

[38] N. Radoev, M. Tremblay, M. Gagnon, and A. Zouaq, "AMAL: Answering French natural language questions using DBpedia," in *Semantic Web Evaluation Challenge*. Cham, Switzerland: Springer, 2017, pp. 90–105.

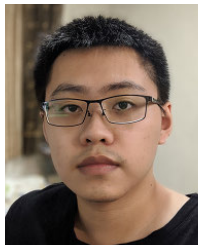
[39] A. Ratnaparkhi, "A maximum entropy part-of-speech tagging," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Sep. 2016, pp. 133–142.

[40] M. A. Smadi, I. A. Dalabih, Y. Jararweh, and P. Juola, "Leveraging linked open data to automatically answer Arabic questions," *IEEE Access*, vol. 7, pp. 177122–177136, 2019.

[41] F. M. Suchanek, G. Kasneci, and G. Weikum, "YAGO: A large ontology from Wikipedia and Wordnet," *J. Web Semantics*, vol. 6, no. 3, pp. 203–217, 2008.
[42] P. N. Tran and D. T. Nguyen, "Mapping expansion of natural language entities to DBpedia's components for querying linked data," in *Proc. 9th Int. Conf. Ubiquitous Inf. Manage. Commun.*, Jan. 2015.
[43] C. Unger, L. Bühmann, J. Lehmann, A. C. Ngonga Ngomo, D. Gerber, and P. Cimiano, "Template-based question answering over RDF data," in *Proc. 21st Int. Conf. World Wide Web*, Lyon, France, Apr. 2012, pp. 639–648.
[44] R. Usbeck, A. C. N. Ngomo, L. Bühmann, and C. Unger, "HAWK–hybrid question answering using linked data," in *Proc. Eur. Semantic Web Conf.*, Portoroz, Slovenia: Springer, Jun. 2015, pp. 353–368.
[45] R. Usbeck, A. C. N. Ngomo, F. Conrads, M. Röder, and G. Napolitano, "8th challenge on question answering over linked data (QALD-8)," in *Proc. CEUR Workshop*, Oct. 2018.
[46] K. Xu, Y. Feng, and D. Zhao, "Answering natural language questions via phrasal semantic parsing," in *Natural Language Processing and Chinese Computing*. Berlin, Germany: Springer, 2014, pp. 333–344.
[47] R. Yus, C. Bobed, and E. Mena, "A knowledge-based approach to enhance provision of location-based services in wireless environments," *IEEE Access*, vol. 8, pp. 80030–80048, 2020.
[48] L. L. Zou, R. H. J. X. W. W. Huang Wang Yu He, and D. Zhao, "Natural language question answering over RDF: A graph data driven approach," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Snowbird, UT, USA, Jun. 2014, pp. 313–324.

YI-HUI CHEN (Member, IEEE) received the Ph.D. degree in computer science and information engineering from National Chung Cheng University. Later on, she worked at Academia Sinica as a Postdoctoral Fellow. Later, she worked as a Research Scientist at the IBM's Taiwan Collaboratory Research Center. After that, she worked at the Department of M-Commerce and Multimedia Applications, Asia University. She is currently an Associate Professor at the Department of Information Management, Chang Gung University. Her research interests include data mining, semantic analysis, and multimedia security.

ERIC JUI-LIN LU (Member, IEEE) received the Ph.D. degree in computer science from the Missouri University of Science and Technology (formerly University of Missouri-Rolla), USA, in 1996. He is currently a Professor with National Chung Hsing University. His research interests include machine learning, natural language processing, and semantic web.



TING-AN OU received the M.S. degree from the Department of Management Information Systems, National Chung Hsing University. His research interests include semantic web and natural language processing.