International Journal on Advanced Science Engineering Information Technology

Vol.14 (2024) No. 4
ISSN: 2088-5334

Automated SPARQL Template for Flexible Question Answering

Dewi Wardani [a,*], Andreas Wijaya [b], Ardhi Wijayanto [b], Maria Ulfah Siregar [c], Yessi Yunitasari [d]

[a] *Department of Data Science, Universitas Sebelas Maret, Surakarta, Central Java, Indonesia*
[b] *Department of Informatics, Universitas Sebelas Maret, Surakarta, Central Java, Indonesia*
[c] *Department of Informatics, UIN Sunan Kalijaga, Yogyakarta, Indonesia*
[d] *Department of Information Technology, Universitas PGRI Madiun, East Java, Indonesia*
[*] *Corresponding author: dww_ok@uns.ac.id*

***Abstract*—The knowledge bases required the query language SPARQL, which consists of subject, property, and object. SPARQL is a structured query language and is difficult to understand. That issue becomes a problem in natural language processing queries. One situation in question answering is how to translate natural language into a structural SPARQL. This work aims to develop an automated SPARQL template algorithm regardless of the pattern structure of the query triples. It provides a more varied SPARQL query for data retrieval named Flexible SPARQL. This approach initially lies in combining elements of RDF with basic techniques of natural language processing to generate a template of SPARQL. In this work, the approach to making automatic templates is proposed without regard to the pattern of the triple structure or the location of the subject and object. Template-based research that exists today still uses rules to determine the position of subjects, objects, and properties in the SPARQL structure. Therefore, this work used the QALD 7 question set and DBpedia dataset. The previous systems utilized the same questions and data sets. Despite the simple proposed approaches that do not use complex, sophisticated techniques, they have shown promising results compared to the previous systems. The accuracy result from 215 questions is 73% and micro-Recall 0.701, micro-Precision 0.664, micro-F-Measure 0.682, macro-Recall 0.711, macro-Precision 0.592, macro-F-Measure 0.646. Overall, the Flexible SPARQL system has higher results on several measurements that define a promising approach. However, it's important to note that Flexible SPARQL generally tends to fail at generating complex SPARQL, which is a limitation of the system.**

***Keywords*— Flexible query; template SPARQL; question answering; knowledge base**

Manuscript received 15 Oct. 2020; revised 29 Jan. 2021; accepted 2 Feb. 2021. Date of publication 31 Aug. 2024.

I. INTRODUCTION

Question Answering (QA). QA has been studied since the 60s. In recent work, the question-and-answer system aims to find information from a dataset knowledge base (KB) using language that is easy to understand for non-professionals [1] [2] [3]. In question answering, translating the natural language queries into a structural SPARQL is one problem that is much in demand. One approach to overcome this issue is to utilize a template to translate question sentences into a SPARQL [4] [5]. A template has a role in determining the answer to a question. Previous work has overcome this problem, but the template was created manually [6], [7]. Different approaches were introduced to some of the QA systems, such as Aqualog, Power-Aqua, NLP-Reduce, and FREyA or the other recent QA systems [8], [9], [10], [11].

Existing templates are still defined directly for the location of subjects, properties, and objects. A simple pattern in a query consisting of subjects, properties, and objects in SPARQL is called a single triple pattern [12]. The question's positioning structure must match the knowledge base pattern to retrieve information. Sometimes, the query structure needs to satisfy the knowledge base structure. Hence, the query cannot obtain the required information. SPIN can used to help SPARQL formulate complex query [13], [14].

The typical tasks in the Semantic Web area use default SPARQL [15] for querying. Some work related to SPARQL templates has been done before. The QA system using an ontology can give meaning to the question [16]. Users prefer using regular question sentences instead of keywords for information retrieval [17]. In works by [18], and [19], the system could translate question sentences in multiple languages into query form. Then, it is used to retrieve information on knowledge. The work focuses on translating simple questions while more complex one's progress. The query structure for retrieving answers must have a pattern against the knowledge bases. The flexible queries can retrieve

information on datasets without knowing the structure used on knowledge bases [20].

There are two components in the work [21]. They are the analysis of questions and answer retrieval. Analysis of questions is according to grammar. Answer retrieval interprets questions against the datasets. Nguyen et al. [22] analyzed several words, such as *"In Berlin were born which actors?"* and taking the word *"Berlin, actors, born in"* as a keyword can provide information on the interpretation of questions with possible queries. The work's results [1] managed to create a system with a template that follows the structure of questions. Unfortunately, it cannot return the information for questions that lead to a knowledge base such as DBpedia. The research concerning converting natural language to SPARQL or vice versa is still attractive [23]. Recent work introducing SPARQL Template proposed a complex approach [16] [24]. Mainly, one works on geospatial data [25].

This work's idea is to create a template that takes freedom in preparing its structure. If the query generally has pattern S-P-O, it also can be arranged in pattern O-P-S. Therefore, the template will be more varied and can be used to retrieve information on knowledge bases without knowing the structure of the knowledge base. This work aims to compile an automated SPARQL template algorithm to create flexible queries. Instead of using a complex approach, this work will utilize a more straightforward method. This work uses the Automatic SPARQL Template to create flexible queries based on the role of each element in triple. The aim is to contribute to this work. This idea was also influenced by the idea that the triple's predicate is the RDF triple's pivot [26]Therefore, the work will focus on the subject or object. The template obtained retrieves information from the knowledge base based on the question.

II. Materials and Method

A. Preprocessing

Preprocessing is a stage in which data is prepared in text and question sentences are used or processed. In preprocessing, there are two stages:

1) POS Tagger: It is the stage where data in the form of sentences are identified and labeled on each word or phrase by type according to the hyphenation or token granting that has been done before [27]. The tagging process aims to break down the labels on each word in the text. The obtained tags can then be used as a term to search for words on DBpedia. Below are examples of tagging results.

How_WRB high_JJ is_VBZ the_DT lighthouse_NNP in_IN Colombo_NNP ?_.
List_NN all_PDT the_DT musicals_NNS with_IN music_NN by_IN Elton_NNP John_NNP ._.
How_WRB many_JJ seats_NNS does_VBZ the_DT home_NN stadium_NN of_IN FC_NNP Porto_NNP have_VB ?_.
Who_WP was_VBD married_VBN to_TO president_NN Chirac_NNP ?_.

2) Filtering: It is a stage for retrieving some words that have already earned label tags. The taken word is a word that has a NN tag. This process is required because the retrieved word will be the term to be used in the SPARQL query. The filtering process aims to retrieve the word that will be the term of a query. NN tags are the key to finding words in DBpedia knowledge bases. Algorithm 1 shows the pseudo-code in obtaining terms within the Filtering process.

Algorithm 1. The obtaining term

```
Input: Tag (Array)
Output: Term (Array)
Step:
begin
        for(tag < size(tags))
                if(jumlah NNP == 0 && term != ‘’)
                        check term the on DBpedia
                if(hasil != null)
                        save term
                elseif(jumlah NNP <= 1)
                        if(tag == NN || IN || FW || CC)
                                term += tagval
                        elseif(tag == NNP)
                                term += tagval
                                check term the on DBpedia
                if(hasil != null)
                save term
end
```

B. Classifying Questions

The classifying stage of a question is where the type of question categorizes the data. This stage is necessary because to run a query from the knowledge base must be known the kind of information retrieved on each query that is run must also provide accurate information on this system. Classifying questions aims to get the kind of question. The question type specifies the syntax to be used in a query. Here is an example of classifying a question.

Query: *Is James Bond married?*
SPARQL: *ASK WHERE {res:James_Bond ont:spouse ?x .}*

In the above example, the question points to a boolean type. So, the syntax used is *ASK*. The formed query will have a count syntax if the question leads to numbers. Here is an example of the query.

Query: *How many languages are spoken in Turkmenistan?*
SPARQL: *SELECT (COUNT(DISTINCT ?x) as ?c) WHERE { res:Turkmenistan ont:language ?x .}*

C. Generating SPARQL Template.

The SPARQL template creation process is a stage for managing the results of filtering and the results of classifying questions into a SPARQL template. The creation of SPARQL needs to pay attention to the existing structure so that each term used can occupy the position of the subject, property, or object. Queries are formed by performing simple permutations for simple queries in the form of query triples.

The result of classifying a question is used to give a syntax to a query such as *ASK* or *COUNT*. In this process, one query can generate many varied queries. Many queries depend on how many terms are used. Each term taken is only sometimes used because only the terms listed on DBpedia are used. The process of creating SPARQL templates, in general, consists of DBpedia search and form classification. The SPARQL template process aims to manage the results of filtering and the results of classifying questions into a SPARQL template.

This process consists of two stages, DBpedia search and classification of forms.

1) DBpedia Search: DBpedia search aims to find relations from filtered words. The DBpedia search section searches for properties on DBpedia by term. The property is used for query formation because queries cannot be formed only with terms. Algorithm 2 shows the pseudo-code in obtaining property. The following are examples of properties that were successfully obtained:

Query: *How high is the lighthouse in Colombo?*
Tag: *How_WRB high_JJ is_VBZ the_DT lighthouse_NNP in_IN Colombo_NNP ?*
Term: lighthouse in Colombo
Term DBpedia:
http:dbpedia.org/resource/Colombo_Lighthouse
Property: http:dbpedia.org/ontology/height;
http:dbpedia.org/ontology/weight
SPARQL:
SELECT DISTINCT ?x WHERE { res:Colombo_Lighthouse ont:height ?x .}. Result: 15.0
SELECT DISTINCT ?x WHERE { ?x ont:height res:Colombo_Lighthouse .}. Result: --
SELECT DISTINCT ?x WHERE { res:Colombo_Lighthouse ont:weight ?x .}. Result: --
SELECT DISTINCT ?x WHERE { ?x ont:weight res:Colombo_Lighthouse .}. Result: --

Therefore, for the resource Colombo_lighthouse, two properties are obtained, so the needed information is obtained. *The height of Colombo_Lighthouse is 15.0.* However, terms are only sometimes found in DBpedia knowledge and retrieval of incorrect terms can result in incorrect queries. In Figure 1 and Figure 2, both terms Lighthouse and Colombo are found in DBpedia, but the formed query does not return answers. This is because the term taken does not match the question, so it does not get correct results from the obtained SPARQL.

Algorithm 2. The obtaining property/predicate
Input: Tag (Array) , Term (Array)
Output: Property (Array)
Step :
begin
for(tag < size(tags))
    if(tag == NN || JJ && not in terms)
        for (term < size terms)
            Property = get property DBpedia (term)
            if(tagval == property)
            save property
end

2) Classification of Query's Form: The classification aims to determine the triple shape that corresponds to the question. Although the obtained query has a free or flexible structure, the obtained triple shape must still be adjusted. The incorrect triple shape will not return the information or did not get different information—classification of forms obtained based on the number of terms and properties obtained. Algorithm 3 shows the pseudo-code in obtaining the query. In the example, the second question results in a longer query than the first. The query on the second question has two triple levels. An example of classification's shape is as below:

Query 1: When was the Battle of Gettysburg?
Term: http://dbpedia.org/resource/Battle_of_Gettysburg
Predicate: http://dbpedia.org/property/date
SPARQL:
SELECT DISTINCT ?x WHERE { res:Battle_of_Gettysburg pro:date ?x .}
SELECT DISTINCT ?x WHERE { ?x pro:date res:Battle_of_Gettysburg .}

Query 2: How many seats does the home stadium of FC Porto have?
Term: http://dbpedia.org/resource/FC_Porto
Predicate: http://dbpedia.org/ontology/seatingCapacity;
http//dbpedia.org/ontology/ground;
http://dbpedia.org/property/owner;
http://dbpedia.org/ontology/Stadium
SPARQL:
SELECT DISTINCT ?x WHERE { res:FC_Porto ont:ground ?y . ?y ont:seatingCapacity ?x . }
SELECT (COUNT(DISTINCT ?x) as ?c) WHERE { res:FC_Porto ont:seatingCapacity ?y . ?y ont:seatingCapacity ?x . }
SELECT (COUNT(DISTINCT ?x) as ?c) WHERE { res:FC_Porto ont:seatingCapacity ?y . ?y ont:ground ?x . }
SELECT DISTINCT ?x WHERE { res:FC_Porto ont:Stadium ?y . ?y ont:seatingCapacity ?x .}

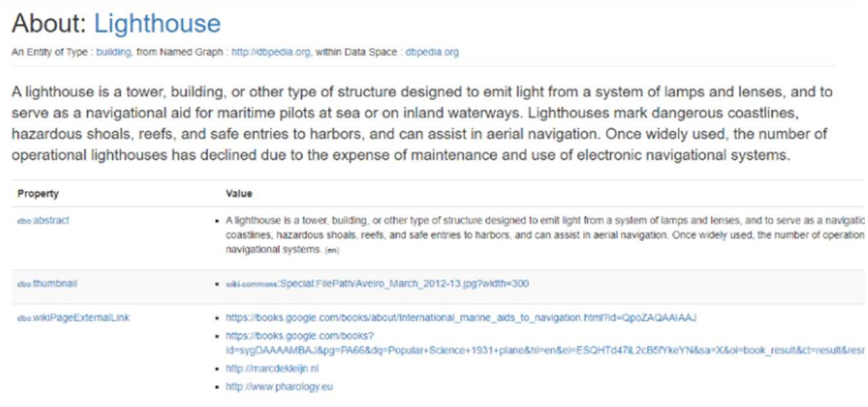

Fig. 1 The example DBpedia's resource explains the Lighthouse.

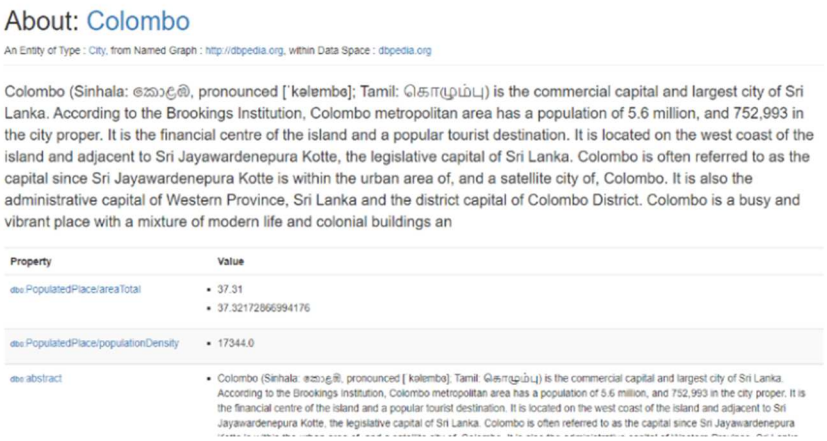

Fig. 2 The example of DBpedia's resource explains Colombo.

Fig. 3 The interface of Template SPARQL's result

When retrieving answers over knowledge bases, the query structure should be the same as the structure in the knowledge bases. However, the same answer can be obtained with different query structures. Some answers can be generated from two different queries. Here is an example:

Query: Who was the wife of U.S. president Lincoln?
SPARQL:
SELECT DISTINCT ?x WHERE { res:Abraham_Lincoln ont:spouse ?x .}. Result: Mary_Todd_Lincoln.
SELECT DISTINCT ?x WHERE { ?x ont:spouse res:Abraham_Lincoln .}. Result: Mary_Todd_Lincoln.
Both above queries return the same answer.

Algorithm 3. The obtaining query
Input: Item (Term, Property)
Output: Query
Step:
begin
run triple(item, triple())
    if(empty item)
        save(triple)
    else
        for(item >=0)
        newitem = item
        newtriple = triple
        list(foo) = arrayshift(newitem,i)
        arrayunshift(newtriple,foo)
        triple(Newitem, Newtriple)
    for (size triple as one)
        query = pertanyaan + one
        run query
    for (size triple as two)
        query += two
        run query
    if(term >=2 && property >= 3)
        for (size triple as three)
        query += two
        run query
end

III. Result and Discussion

A. The Dataset

The data collected consists of text tagger data in English. The word set data is in English data to be created SPARQL query and data in DBpedia knowledge base. The data text tagger used is a language model with an English tagger compressed with a file size of 15,437 KB. The word set data consists of 550.00 words that will be used to retrieve words with the same meaning as different writings. Data were taken from DBpedia. It is a knowledge base describing a word, property as a verb and ontology defined in DBpedia mapping. DBpedia has 5.500.000 data consisting of 1.500.000 people data, 840.000 place data, 496.000 artworks, 286.000 organizational data, 306.000 species, 58.000 plants, and 6.000 diseases. The data will be composed into SPARQL. The data can have a structured pattern (subjects, predicates, objects) or unstructured data. The data consist of 215 questions with a file size of 486 KB in JSON format from the QALD 7 dataset. The example of QALD-7's dataset (in JSON) is a snippet below:

"questions": [{
"id": "0",
"answertype": "date",
"aggregation": false,
"onlydbo": true,
"hybrid": false,
"question": [{
"language": "en", "string": "When was the Battle of Gettysburg?",
"keywords": "Battle of Gettysburg"
},{
"language": "pt_BR",
"string": "Quando foi a batalha de Gettysburg?",
"keywords": "batalha de Gettysburg"
},{

```
"language": "de",
"string": "Wann fand die Schlacht von Gettysburg statt?",
"keywords": "Schlacht von Gettysburg"
},{
"language": "es",
"string": "¿Cuándo tuvo lugar la batalla de Gettysburg?",
"keywords": "batalla de Gettysburg"
},{
"language": "it",
"string": "Quando ha avuto luogo la battaglia di Gettysburg?",
"keywords": "battaglia di Gettysburg"
},{
"language": "fr",
"string": "Quand se déroula la bataille de Gettysburg?",
"keywords": "bataille de Gettysburg, quand"
},{
"language": "nl",
"string": "Wanneer was de Slag bij Gettysburg?",
"keywords": "Slag bij Gettysburg"
},{
"language": "hi_IN",
"string": "गेटिसबर्ग का युद्ध कब हुआ था?",
"keywords": "गेटिसबर्ग का युद्ध"
}],
"query": {
"sparql": "PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX res: <http://dbpedia.org/resource/> SELECT
DISTINCT ?date WHERE { res:Battle_of_Gettysburg
dbo:date ?date .}"
},
"answers": [{
"head": {
"vars": [
"date"
]
},
"results": {
"bindings": [{
"date": {
        "type": "literal",
        "value": "07--01"
}}]}}]},
```


B. The Result

In this work, the text data of questions in English was structured or unstructured. Question data was obtained from QALD 7, as explained in the previous part. Evaluation is based on Precision, Recall, and F-Measure. The Template SPARQL, explained in the last part, has been implemented and developed. Figure 3 shows a part of the interface of the system. The example result of the generated query can be seen in Figure 3. It is seen that two queries get information from the question:

Query: Who is the host of the BBC Wildlife Specials?
SPARQL:
SELECT DISTINCT ?x WHERE {res:BBC_Wildlife_Specials ont:numberOfEpisodes ?x.}
Result 1: *29*
SELECT DISTINCT ?x WHERE {res:BBC_Wildlife_Specials ont:presenter ?x.}
Result 2: *http://dbpedia.org/resource/David_Attenborough*
Gold answer:
http://dbpedia.org/resource/David_Attenborough

Therefore, result 1 is wrong, and result 2 is correct. Hence, the Precision is 0.5, and the Recall is 1.

The initial evaluation stage is carried out by comparing the answers with the results from QALD 7. This is done to determine the correct answer generated from the SPARQL template because QALD 7 already has the gold answer. In some questions, the gold answers are obtained from DBpedia 2020. The evaluation is carried out by calculating Precision, Recall and F-Measure. The results of generating SPARQL queries and calculating accuracy, Recall and Precision for each question are in the appendix. The other example is as below:

Query: When was the Battle of Gettysburg?
SPARQL:
SELECT DISTINCT ?x WHERE {res:Battle_of_Gettysburg pro:date ?x.}. Result 1: --07-01
SELECT DISTINCT ?x WHERE {?x pro:date res:Battle_of_Gettysburg.}. Result 2: --
Gold answer: 1863-07-03
Gold answer from DBpedia 2020: --07-01

The above query is an example to explain this situation for a few questions. The following is an example of DBpedia 2020 data. In DBpedia 2020, the date property on the Battle of Gettysburg resource has the information --07-01. In addition to testing using questions from QALD 7, in the evaluation, there are also several questions outside of QALD 7. Here are examples of questions outside of QALD 7. Comparison of the SPARQL template system with other systems WDAqua [28] and ganswer2 [29], shown in Table 1. These two systems used the same question dataset.

TABLE I
THE COMPARING RESULT OF EACH SYSTEM WITH TEMPLATE SPARQL

The Measurement	WDAqua	ganswer2	Flexible SPARQL
Micro-P	-	0.113	0.501
Micro-R	-	0.561	0.682
Micro-FM	-	0.189	0.577
Macro-P	0.490	0.557	0.404
Macro-R	0.540	0.592	0.703
Macro-FM	0.510	0.556	0.513

Table 2 shows the example of the detailed result of Flexible SPARQL.

TABLE II
THE EXAMPLE OF OBTAINED SCORE OF FLEXIBLE SPARQL

No	Question	P	R	F
1	When was the Battle of Gettysburg?	1	1	1
2	How high is the lighthouse in Colombo?	1	1	1
3	Who was the wife of U.S. president Lincoln?	0.25	1	0.4
4	Who is the host of the BBC Wildlife Specials?	0.285	1	0.44
5	How much did Pulp Fiction cost?	0.5	1	0.667
6	Who developed Slack?	0	0	0

No	Question	P	R	F
7	When did Operation Overlord commence?	0.5	1	0.667
8	Who painted The Storm on the Sea of Galilee?	0	0	0
9	Which museum exhibits The Scream by Munch?	1	1	1
10	Is James Bond married?	0.818	1	0.9
11	Who is the mayor of Paris?	0.667	1	0.8
12	Which awards did Douglas Hofstadter win?	0.5	1	0.667
13	How much did the Lego Movie cost?	0.33	1	0.5
14	How many languages are spoken in Turkmenistan?	1	1	1
15	Is Christian Bale starring in Batman Begins?	0.833	1	0.909

Overall, the Flexible SPARQL has higher results than the WDAqua system and on the ganswer2 micro value. Macro-Recall gets a higher value, although macro precision is still under other systems. In general, the results obtained are influenced by several factors: Tagging process on questions, keyword retrieval for questions, determination of question types, and the form of queries to be created.

The Flexible SPARQL does not cover complex queries. For example, "Give me all professional skateboarders from Sweden." This is because the determination of complex or straightforward queries has yet to be carried out. If complex queries are added without any such determination, the query variations will be many, and the results obtained will also be further from the actual results.

There are two steps in ganswer2: an online step and an offline phase. Using a graph mining method, the offline step creates data based on the relationship between the steps and the existing predicates. The online stage is used to understand and evaluate queries depending on the data collected. To express the question's meaning as explained in [30], ganswer2 builds a query graph. To create SPARQL, WDAqua presented a rule-based combinatorial. They translate natural language into SPARQL using the used knowledge base rather than machine learning algorithms. Because of the limitations of the given criteria, the number of SPARQL queries that can be generated is limited. The acquired SPARQL is said to include no more than two triple patterns. The resulting query is restricted to the *COUNT* operator. In WDAqua, adding new operators to the resulting query will necessitate much effort in design and transformation rules as narrated in [31]. The query must still be retrievable using the SPARQL template for more complex searches. A problematic question like the one below is an example:

Query: Give me all professional skateboarders from Sweden.
SPARQL:

```
SELECT DISTINCT ?uri WHERE {
  ?uri dbo:occupation res:Skateboarding .
  { ?uri dbo:birthPlace res:Sweden .}
  UNION
  { ?uri dbo:birthPlace ?place .
    ?place dbo:country res:Sweden .}
  }}
```

Result:
http://DBpedia.org/resource/Ali_Boulala,\\http://DBpedia.org/resource/Tony_Magnusson

The Flexible SPARQL does not cover complex queries because complex or straightforward queries have not yet been determined. If complex queries are added without any such determination, there will be many query variations, and the results obtained will also be further from the actual results. In general, the results obtained are influenced by several factors, including the process of tagging the question, taking terms for the question, determining the type of question, and determining the form of the query to be made.

There are several notes in the preparation of the automatic SPARQL template algorithm, as below:

- The process of determining words, especially in retrieving resources in DBpedia, because there are words that are different from the results of taking terms. Some words are not found in the property. Determining this term is the most influencing obstacle because it affects the form of the query and the combination of triple queries.
- Classification of question types is still not accurate
- Triple-form classification still cannot be used to form more complex queries.
- The query made does not have a filter, even though it has a result. The result does not follow the existing question, which causes the Precision value to be below.

IV. Conclusions

For building Flexible SPARQL, this work presented an automated SPARQL template algorithm. The work's final product is a SPARQL query that can be used to get data from the DBpedia knowledge base. The accuracy percentage result obtained in this work was 73%. The results for Precision and Recall are Micro-Precision 0.501, Micro-Recall 0.682, Macro-Precision 0.404, and Macro-Recall 0.703. The Precision is lower than the Recall result because the number of answers exceeds the actual answer. The number of answers is generated because the queries created are also more diverse. The first future work is to investigate other methods in determining keywords. The following work is to investigate the approach to classing questions. Therefore, each question can be well-identified. The other important work to be done soon is improving the POS Tagging performance for better results. The other urgent future work is how this Template SPARQL solves complex queries, such as using UNION. Another future work is how to implement the template for a new model, APRDF [32].

Acknowledgment

The authors thank the funding of this research from Universitas Sebelas Maret.

References

[1] 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, Apr. 2020, doi: 10.3233/sw-190343.

[2] N. D. To and M. Reformat, "Question-Answering System with Linguistic Terms over RDF Knowledge Graphs," *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct. 2020, doi: 10.1109/smc42975.2020.9282949.

[3] S. Wang, J. Jiao, and X. Zhang, “A Semantic Similarity-based Subgraph Matching Method for Improving Question Answering over RDF,” *Companion Proceedings of the Web Conference 2020*, Apr. 2020, doi: 10.1145/3366424.3382698.

[4] A. Saxena, A. Tripathi, and P. Talukdar, “Improving Multi-hop Question Answering over Knowledge Graphs using Knowledge Base Embeddings,” *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* 2020, doi:10.18653/v1/2020.acl-main.412.

[5] Y. Qiu, Y. Wang, X. Jin, and K. Zhang, “Stepwise Reasoning for Multi-Relation Question Answering over Knowledge Graph with Weak Supervision,” *Proceedings of the 13th International Conference on Web Search and Data Mining*, Jan. 2020, doi:10.1145/3336191.3371812.

[6] R. G. Athreya, S. K. Bansal, A.-C. N. Ngomo, and R. Usbeck, “Template-based Question Answering using Recursive Neural Networks,” 2021 IEEE 15th International Conference on Semantic Computing (ICSC), Jan. 2021, doi: 10.1109/icsc50631.2021.00041.

[7] A. K. Dileep, A. Mishra, R. Mehta, S. Uppal, J. Chakraborty, and S. K. Bansal, “Template-based Question Answering analysis on the LC-QuAD2.0 Dataset,” 2021 IEEE 15th International Conference on Semantic Computing (ICSC), Jan. 2021, doi:10.1109/icsc50631.2021.00079.

[8] X. Lin and Y. Zhang, “Knowledge Graph Post-Processing for QA Systems,” *2021 7th International Conference on Computing and Artificial Intelligence*, Apr. 2021, doi: 10.1145/3467707.3467758.

[9] T. Souza Costa, S. Gottschalk, and E. Demidova, “Event-QA,” *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, Oct. 2020, doi:10.1145/3340531.3412760.

[10] M. Del Tredici, G. Barlacchi, X. Shen, W. Cheng, and A. de Gispert, “Question Rewriting for Open-Domain Conversational QA,” *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, Oct. 2021, doi:10.1145/3459637.3482164.

[11] P. Qin et al., “Unified QA-aware Knowledge Graph Generation Based on Multi-modal Modeling,” *Proceedings of the 30th ACM International Conference on Multimedia*, Oct. 2022, doi:10.1145/3503161.3551604.

[12] W. Ali, M. Saleem, B. Yao, A. Hogan, and A.-C. N. Ngomo, “A survey of RDF stores & SPARQL engines for querying knowledge graphs,” *The VLDB Journal*, vol. 31, no. 3, pp. 1–26, Nov. 2021, doi:10.1007/s00778-021-00711-3.

[13] D. Wardani and M. Susmawati, “SESS: Utilization of SPIN for Ethnomedicine Semantic Search,” *Proceedings of the 2022 International Conference on Computer, Control, Informatics and Its Applications*, Nov. 2022, doi: 10.1145/3575882.3575912.

[14] K. S. Gan, P. Anthony, K. O. Chin, and A. R. Hamdan, “Enforcing Social Semantic in FIPA-ACL Using SPIN,” *Smart Innovation, Systems and Technologies*, pp. 3–13, Jun. 2019, doi: 10.1007/978-981-13-8679-4_1.

[15] C. Wang and X. Zhang, “Q-BERT: A BERT-based Framework for Computing SPARQL Similarity in Natural Language,” *Companion Proceedings of the Web Conference 2020*, Apr. 2020, doi:10.1145/3366424.3382699.

[16] P. Kaur, and P. Nand, "Towards Transparent Governance by Unifying Open Data," *IAENG International Journal of Computer Science*, vol. 48, no.4, pp986-1004, 2021

[17] A. Ben Ayed, I. Biskri, and J.-G. Meunier, “An Enhanced Lucene based System for Efficient Document/Information Retrieval,” *Computer Science & Information Technology*, Jul. 2020, doi:10.5121/csit.2020.100913.

[18] Y. Tan, Y. Chen, G. Qi, W. Li, and M. Wang, “MLPQ: A Dataset for Path Question Answering over Multilingual Knowledge Graphs,” *Big Data Research*, vol. 32, p. 100381, May 2023, doi:10.1016/j.bdr.2023.100381.

[19] A. Perevalov, D. Diefenbach, R. Usbeck, and A. Both, “QALD-9-plus: A Multilingual Dataset for Question Answering over DBpedia and Wikidata Translated by Native Speakers,” *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*, Jan. 2022, doi: 10.1109/icsc52841.2022.00045.

[20] R. Frosini, A. Poulovassilis, P. T. Wood, and A. Calí, “Optimisation Techniques for Flexible SPARQL Queries,” *ACM Transactions on the Web*, vol. 16, no. 4, pp. 1–44, Nov. 2022, doi: 10.1145/3532855.

[21] A. Calì, R. Frosini, A. Poulovassilis, and P. T. Wood, “Flexible Querying for SPARQL,” On the Move to Meaningful Internet Systems: OTM 2014 Conferences, pp. 473–490, 2014, doi: 10.1007/978-3-662-45563-0_28.

[22] D. Q. Nguyen, D. Q. Nguyen, and S. B. Pham, “Ripple Down Rules for question answering,” *Semantic Web*, vol. 8, no. 4, pp. 511–532, Jan. 2017, doi: 10.3233/sw-150204.

[23] E. Adhim and D. Wardani, “Improving the Result of Question Answering System with Semantic Similarity Method Based on Hierarchy in Ontology,” *Proceedings of the 2021 International Conference on Computer, Control, Informatics and Its Applications*, Oct. 2021, doi: 10.1145/3489088.3489095.

[24] X. Yin, D. Gromann, and S. Rudolph, “Neural machine translating from natural language to SPARQL,” *Future Generation Computer Systems*, vol. 117, pp. 510–519, Apr. 2021, doi:10.1016/j.future.2020.12.013.

[25] D. Punjani and E. Tsalapati, “Question Answering Engines for Geospatial Knowledge Graphs,” *Geospatial Data Science*, pp. 257–282, Jun. 2023, doi: 10.1145/3581906.3581922.

[26] M. Bakhshi, M. Nematbakhsh, M. Mohsenzadeh, and A. M. Rahmani, “Data-driven construction of SPARQL queries by approximate question graph alignment in question answering over knowledge graphs,” *Expert Systems with Applications*, vol. 146, p. 113205, May 2020, doi: 10.1016/j.eswa.2020.113205.

[27] N. Bölücü and B. Can, “A Cascaded Unsupervised Model for PoS Tagging,” *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 20, no. 1, pp. 1–23, Jan. 2021, doi:10.1145/3447759.

[28] D. Diefenbach, K. Singh, and P. Maret, “WDAqua-core1: A Question Answering service for RDF Knowledge Bases,” *Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW ’18*, 2018, doi: 10.1145/3184558.3191541.

[29] R. Huang and L. Zou, “Natural language question answering over RDF data,” *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, Jun. 2013, doi:10.1145/2463676.2463725.

[30] Y.H. Chen, E. J.-L. Lu, and T.-A. Ou, “Intelligent SPARQL Query Generation for Natural Language Processing Systems,” *IEEE Access*, vol. 9, pp. 158638–158650, 2021, doi: 10.1109/access.2021.3130667.

[31] S. Liang, K. Stockinger, T. M. de Farias, M. Anisimova, and M. Gil, “Querying knowledge graphs in natural language,” *Journal of Big Data*, vol. 8, no. 1, Jan. 2021, doi: 10.1186/s40537-020-00383-w.

[32] D. Wardani, “Complete W3C-Semantic’s Interpretations of AP-RDF,” *IAENG International Journal of Computer Science*, vol. 49, no. 3, 2022.