SPARQL Generation with an NMT-based Approach

Jia-Huei Lin and Eric Jui-Lin Lu*

National Chung Hsing University, Taichung, Taiwan (R.O.C.)
E-mail: jllu@nchu.edu.tw
**Corresponding Author*

Received 29 September 2021; Accepted 13 March 2022;
Publication 30 July 2022

Abstract

SPARQL is a powerful query language which has been widely used in various natural language question answering (QA) systems. As the advances of deep neural networks, Neural Machine Translation (NMT) models are employed to directly translate natural language questions to SPARQL queries in recent years. In this paper, we propose an NMT-based approach with Transformer model to generate SPARQL queries. Transformer model is chosen due to its relatively high efficiency and effectiveness. We design a format to encode a SPARQL query into a simple sequence with only RDF triples reserved. The main purpose of this step is to shorten the sequences and reduce the complexity of the target language. Moreover, we employ entity type tags to further resolve mistranslated problems. The proposed approach is evaluated against three open-domain question answering datasets (QALD-7, QALD-8, and LC-QuAD) on BLEU score and accuracy, and obtains outstanding results (83.49%, 90.13%, and 76.32% on BLEU score, respectively) which considerably outperform all known studies.

Keywords: SPARQL generation, neural machine translation, question answering, transformer.

Journal of Web Engineering, Vol. 21_5, 1471–1490.
doi: 10.13052/jwe1540-9589.2155

1 Introduction

SPARQL is the W3C recommended query language for retrieving data from knowledge graphs (KG) stored in the Resource Description Framework (RDF), a directed and labelled graph data format [1]. Searching for information with SPARQL is significantly more accurate and efficient than using a regular search engine since SPARQL requires an understanding of the syntax and semantics of a natural language and constructs structured queries [2]. Due to its highly powerful capability, SPARQL has already been widely used in various information systems, making it an important and urgent task to automatically generate SPARQL queries.

Question answering over linked data (QALD) is one of the applications of SPARQL [3]. For instance, a well-designed question answering system based on DBpedia (a famous linked data which extracted structured information from Wikipedia [4]) might be able to convert a natural language question 'Where is Fort Knox located?' to its SPARQL query 'SELECT DISTINCT ?ans WHERE {dbr:Fort_Knox dbo:location ?ans.}' and retrieve the exact answer 'Kentucky' for this question from DBpedia. The RDF triple 'dbr:Fort_Knox dbo:location ?ans' in this SPARQL query can be simply interpreted as follows: "The answer is stored under the property 'dbo:location' of the named entity 'Fort Knox' in DBpedia.", where 'Fort Knox' is the subject, 'dbo:location' is the relation, and '?ans' is the object of this RDF triple, as shown in Figure 1.

As the advances of deep neural networks, the encoder-decoder architecture with neural networks has become a de facto approach for solving sequence-to-sequence (seq2seq) tasks as well as building neural machine translation (NMT) models, while models under this architecture are able to handle input and target sequences with variable lengths [5] as shown in Figure 2. In recent years, NMT models are employed not only to translate between two natural languages, but also to directly translate natural language questions to SPARQL queries [2, 3, 6, 7]. Although the previous studies [2, 3, 6, 7] have reached good results in some closed-domain question answering datasets [6, 8], they failed to solve open-domain question answering datasets like LC-QuAD [9], mainly because of the high complexity of SPARQL queries as well as the limited vocabulary size. Take the SPARQL query 'SELECT DISTINCT ?ans WHERE {dbr:Fort_Knox dbo:location ?ans.}' as an example, it is hard for a model to correctly put every single punctuation (e.g., '.' and '{'), variable (e.g., '?ans'), specific SPARQL syntax (e.g., 'SELECT DISTINCT'), and entity (e.g., 'dbo:location') in the right

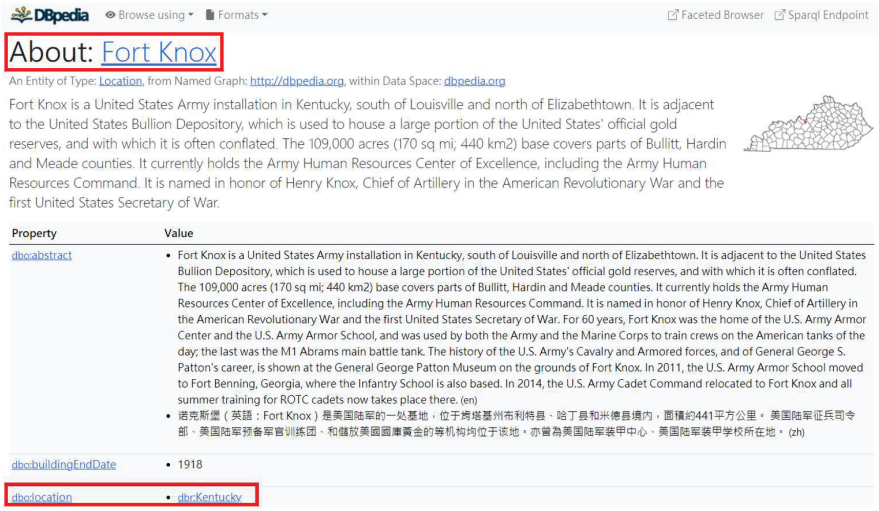

Figure 1 An DBpedia Example. Part of the properties and values of the named entity 'Fort Knox' in DBpedia.

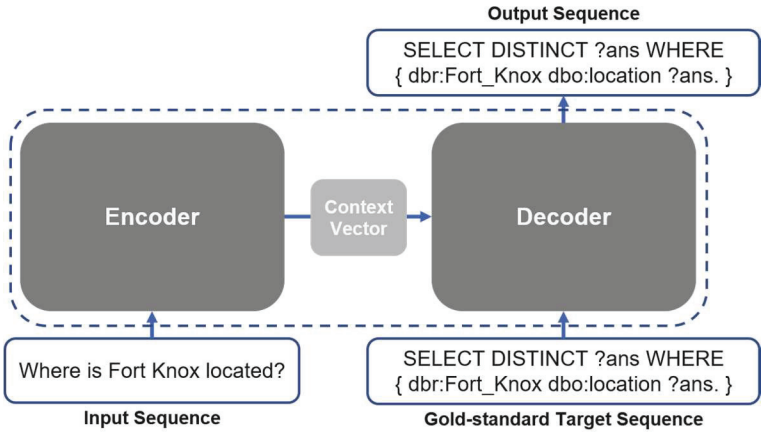

Figure 2 Encoder-decoder Architecture. An example of using a simple NMT model under the encoder-decoder architecture to translate a natural language question to a SPARQL query.

position, not to mention that it is hardly possible for the model to be able to generate unseen entities that do not exist in the training set. These difficulties make it challenging to practically apply NMT models into question answering systems to generate SPARQL by far.

Traditionally, there are two main approaches to automatically generate SPARQL. The first one is to turn natural language questions into an intermediary format (ex. entity type tags), generally with some multiclass classifiers, followed by templates or dependency structure to generate the final SPARQL queries [10–14]. The second one is to rely on the structure of knowledge graph, applying algorithms for subgraph searching to find all possible RDF triples [15]. These approaches are capable of handling some complex SPARQL queries; however, the cost of querying stays at a relatively high level. As for using NMT models to generate SPARQL, Convolutional Sequence to Sequence Model (ConvS2S) [16] and Transformer model [17] are mainly chosen to train the SPARQL generator, and most of the previous studies utilized the 'one to one SPARQL encoding method' to encode SPARQL queries, while it is found that by shortening and simplifying the SPARQL queries can improve the translation result [3, 7].

In [18], we proposed an NMT-based approach with the Transformer model to automatically translate natural language questions to SPARQL queries and presented our preliminary results. A Transformer model is chosen to train our SPARQL generator due to its relatively high efficiency than RNN-based models as well as the effectiveness in various natural language processing tasks [17]. To simplify SPARQL queries and overcome mistranslation problems, which commonly happen in punctuations and specific syntax in SPARQL, we design a new format to encode a SPARQL query, where only RDF triples are reserved as the target language for the NMT model. Moreover, a new NER string-like variable is designed in our approach to reduce the complexity of both the natural language question and SPARQL for named entities. For the input sequences (ie. natural language questions), pre-trained word embedding is utilized to minimize the out-of-vocabulary problem. Another specialty of our presented approach is that we modify translation results by entity type tags [11, 14] to help making up for the limited target vocabulary size.

Experimental results of this approach on the QALD-7 dataset have been published in our preliminary study [18]. The outstanding results encouraged us to pursue further. To systematically study the proposed approach and the architecture of the Transformer model, a complete series of experiments are conducted in this study, including two extra open-domain question answering datasets (QALD-8 and LC-QuAD). LC-QuAD dataset is involved to better compare our approach to the previous ones [2, 3]. Furthermore, QALD-8 dataset is tested not only to benchmark QALD-7, but the QALD series is considered as significantly more complex datasets having richer

characteristic than LC-QuAD [19], which can further prove the effectiveness of our approach. All results are evaluated with both BLEU score and accuracy, where BLEU score presents the correctness of words and phrases appear in the output sequence, and accuracy mainly accentuates whether every single token is in the right position. Our approach achieved outstanding 83.49%, 90.13%, and 76.32% on BLEU score using QALD-7, QALD-8, and LC-QuAD, respectively, while the accuracy score using the LC-QuAD dataset is 72%, considerably outperform all previous studies [2, 3], which only reached about 10% on accuracy.

2 Related Works

2.1 Neural Machine Translation

Deep neural networks under the encoder-decoder architecture have shown to be powerful in not only machine translation but any other sequence to sequence tasks [20]. One of the characteristics of a sequence to sequence task is that the input and target sequences can have different lengths [5]. As shown in Figure 2, in the training phase, the encoder learns and extracts important information from the input sequence into a context vector, where the length of the input sequence is five, while the decoder considers and absorbs information from both the context vector and the gold-standard target sequence, learning how to generate the output sequence with a totally different length from the input sequence. Additionally, in the inference phase, the model chooses from the dictionary of the target language to find the most suitable word for each position. In most of the prevalent approaches, the encoder and decoder were based on recurrent neural networks (RNN) that benefit sequential learning until the ConvS2S model [16] and Transformer model [17] were launched.

Google's Neural Machine Translation system (GNMT) [21] is constructed by a deep LSTM network which consists of 8 encoder and 8 decoder layers with attention mechanism between the bottom layer of the decoder to the top layer of the encoder. ConvS2S [16] is a fully convolutional architecture uses convolutions and gated linear units in its encoder and decoder layers, including attention mechanism as well to capture the relation between input and target language. Transformer model [17] is, on the other hand, the state-of-the-art machine translation model which based solely on attention mechanism, with its encoder and decoder layers are all constructed by self-attention. Both ConvS2S and Transformer models are way more efficient and better exploit the hardware than RNN-based methods [16, 17].

2.2 SPARQL Generation Using NMT Models

Using NMT models in SPARQL generation has been a popular field in recent years. Soru et al. [6] proposed an architecture to translate natural language questions to SPARQL queries with a basic LSTM-based NMT model. In this study [6], SPARQL is encoded into a sequential format that operators, brackets, and URIs are replaced by string-like variables as the target language of the NMT model. For instance, a SPARQL query 'SELECT DISTINCT ?uri WHERE {dbr:Fort_Knox dbo:location ?uri.}' would be encoded into 'select distinct var_uri where brack_open dbr_Fort_Knox dbo_location var_uri sep_dot brack_close'. Results of this approach were evaluated with the Monument Dataset, a closed-domain dataset expanded from the class 'dbo:Monument'. The Monument Dataset is built with templates, meaning that only queries that satisfied the corresponding SPARQL patterns will be fetched from DBpedia, and the named entities are also restricted by specific properties [6], making it considerably less complex than LC-QuAD [9] or any other open-domain datasets. Moreover, a closed-domain question answering dataset seldom runs into out-of-vocabulary problem, since the named entities and property entities are all limited to a specific class. A further research also done by Soru et al. [7] proved that shortening SPARQL sequences and adding direct entity translations can improve the translation results.

Yin et al. [2] utilize eight NMT models, including 6 LSTM-based models, ConvS2S model, and Transformer model, as well as three question answering datasets (The Monument Dataset [6], DBNQA [8], and LC-QuAD [9]) to investigate suitable models for SPARQL generation. ConvS2S outperformed the other 7 models in this task, reaching a 97.12% in the Monument Dataset and a 59.54% in LC-QuAD on BLEU score. The six LSTM-based models used in this study are mainly on the basis of GNMT [21] but with different structure, hyperparameter and attention mechanism; these models, however, perform badly in the SPARQL generation task. As for the Transformer model, it was in the second place and performed well in smaller datasets but failed to handle DBNQA [8] which contains about 900,000 question pairs. In addition, none of the eight models produced one fully correct query in LC-QuAD, where the highest accuracy was only 8% by the ConvS2S model, meaning that there are still defects on these approaches to solve an open-domain and complex dataset.

Rather than directly translate a natural language question to SPARQL, Diomedi and Hogan [3] employ NMT models to produce an appropriate template with generic placeholders for a question, combining the result of

sequence labelling and entity linking to get the final SPARQL query. For instance, the phrase 'Fort Knox' is marked as 'subj' in the question 'Where is Fort Knox located?' in the sequence labelling component, while entities related to 'Fort Knox' found in the entity linking component will be put into the placeholder '<subj>' in the generated template 'SECLCT DISTINCT ?obj WHERE {<subj> dbo:location ?obj.}'. The best BLEU score and accuracy of this approach reached 59.3% and 14.0% respectively in LC-QuAD dataset, showing that this approach can partially fix the out-of-vocabulary problem that the previous studies [2, 6] suffer from; however, it still run into problem in complex questions.

As can be seen, the previous studies [2, 3] which can perfectly deal with closed-domain question answering datasets all fail to handle the LC-QuAD [9] dataset, where the highest accuracy they reached is only about 60%. On the contrary, our preliminary study [18] shows an outstanding accuracy (78.07%) using QALD-7 dataset, where the QALD datasets are considered as significantly more complex datasets LC-QuAD [19]. The preliminary experimental results not only show that our approach is feasible to solve open-domain question answering datasets, but also encourage us for further investigation.

3 Our Approach

In this section, the architecture of our presented approach will be introduced in details, separated into training and inference phases.

3.1 The Training Phase

As illustrated in Figure 3, there are mainly three components in our training phase, including the data pre-processing, embedding layer, and the Transformer model. Similar to any other machine translation model, in our approach, the input and target sequences must be turned into an expected format at first, followed by an embedding layer to transform the string-type data to a numeric one, and lastly, a sequence to sequence model, the Transformer model in our case, is utilized to train the pre-processed data. After training, the trained Transformer model turns out to be the SPARQL generator in our inference phase to generate the preliminary SPARQL queries.

Data Pre-processing. To encode a SPARQL query into a sequential format is an essential step when adopting an NMT model for SPARQL

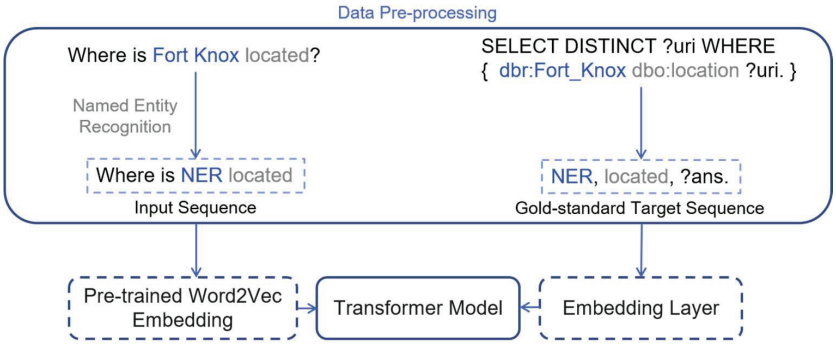

Figure 3 Architecture of Training Phase. The training phase of our approach with an example.

generation [7]. In our data pre-processing component, question-query pairs in the training datasets are artificially turned into a newly designed format. The question-query pairs are the natural language questions and their corresponding SPARQL queries originally from the QALD-7, QALD-8, and Lc-QUAD datasets. We encode a SPARQL query into a simple sequence to shorten and simplify the target language, because it is found in previous studies [3, 6] that the simpler the target language is, the better the translation result will be. Figure 4 shows the comparison among different SPARQL encoding methods using the same Fort Knox example. It is clear that our newly designed format can not only provide the minimum length and the least complexity of the target language (the encoded SPARQL queries), but still sustain sufficient information to answer questions. In our newly designed format, only RDF triples in a SPARQL query are reserved for presenting the query intention, while punctuations and specific syntax in SPARQL are removed since they can be easily added back by some simple post processing.

In our data pre-processing component, for input sequence, named entities are marked with a natural language processing tool such as NLTK NER tagger. If any phrase in the input natural language question is recognized as a named entity, we replace the phrase with a variable 'NER' since named entities are usually proper nouns like the name of a person or a place, which are way too specific and complex for the model to get to 'know' every of them, leading to out-of-vocabulary problem. At the same time, if the recognized named entity is used in the target SPARQL query, the entity will also be replaced with variable 'NER'. As shown in Figure 3, for the question 'Where is Fort Knox located?', it is converted to 'Where is NER located', because 'Fort Knox' was tagged as an NER. Also, the

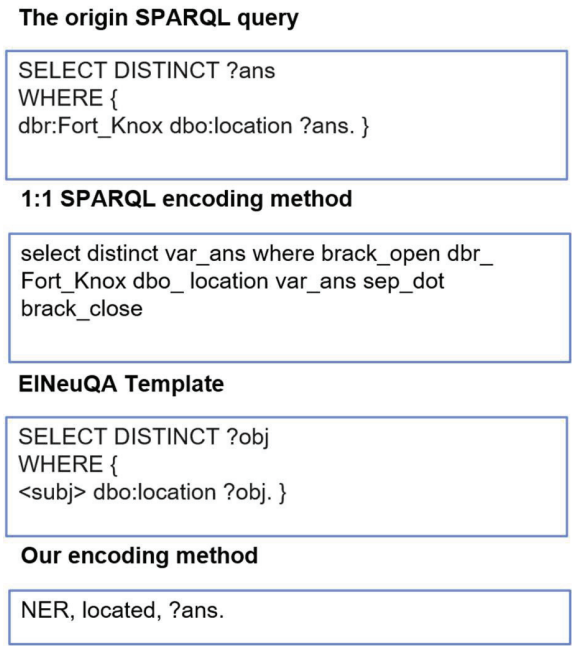

Figure 4 SPARQL Encoding Methods. Comparison among different SPARQL encoding methods [3, 6].

entity 'dbr:Fort_Knox' in the target SPARQL query is converted to NER as well. The design of 'NER' has an extra advantage. If the input sequence contains the word 'USA', the corresponding entity of this word in the target sequence should be 'dbr:United_States'. Because 'USA' is now marked as 'NER', the post processing step can easily convert it to 'dbr:United_States' based on the entity type tag [11, 14]. However, it is very hard for any other translator to convert 'USA' to 'United States' and find the corresponding entity 'dbr:United_States' due to the lexical gap and ambiguity problem.

As for property or class entities in the target sequence, we look up every single entity in the input sequence and find a most suitable word to replace the entity. For example, 'dbo:location' is replaced by 'located'.

Finally, less important tokens in the SPARQL queries are all removed and only RDF triples are reserved, while commas are added between each entity to separate subject, predicate, and object in an RDF triple. After all, only 'NER, located, ?ans.' is reserved as the target language in our approach, which is way more simplified for a machine translation model to learn than the prior methods [2, 3, 6, 7].

Table 1 Comparison of hyperparameters between the original version of Transformer model and the presented one in this paper

Model	Layer Number	Multi-head Number	Learning Rate	Embedding Size	Embedding
Original Transformer	6	8	Noam Scheduler	512	Keras Embedding Layer
The presented Transformer	2	4	0.00015	256	Pre-trained (Word2Vec)

Word Embedding Pre-training. The original Transformer model utilized the Tensorflow's embedding layer in both the input and target language, meaning that only the vocabularies appear in the training set are used to train the word embeddings. In our approach, we do the same thing in our target language (the encoded SPARQL), but pre-trained embedding is employed in our input language to reduce unknown words since our datasets are relatively smaller ones.

However, as introduced previously, our input language is not usual English, but all named entities are replaced by the string 'NER's, such as 'Where is NER located?'. Therefore, we created our own pre-trained embedding for our input language under the following process. Firstly, we downloaded all English articles on Wikipedia (Oct. 2020 version), having approximately 6 million articles, and marked every named entity in these articles with the NLTK's NER tagger. These processed articles are then used to train our own word embeddings with the Gensim's Word2vec model.

The Transformer Model. The SPARQL generator is then trained with the pre-processed data using a Transformer model [17]. In this paper, we focus only on Transformer model due to its relatively high efficiency. The hyperparameter tuning of our presented Transformer model will be described in detail in Section 4.2. After experiments, Table 1 lists the best hyperparameters of our presented model, while pre-trained Word2Vec embedding is employed.

3.2 The Inference Phase

Figure 5 illustrates the complete process of our inference phase with an example. In the following paragraphs, the process will be described accompany with the example 'Where is Fort Knox located?'.

First, named entities in the input sentence will be marked with NLTK NER tagger, just like what we do in the training phase. In this example, 'Fort Knox' will be marked as a named entity, so the input sentence is now

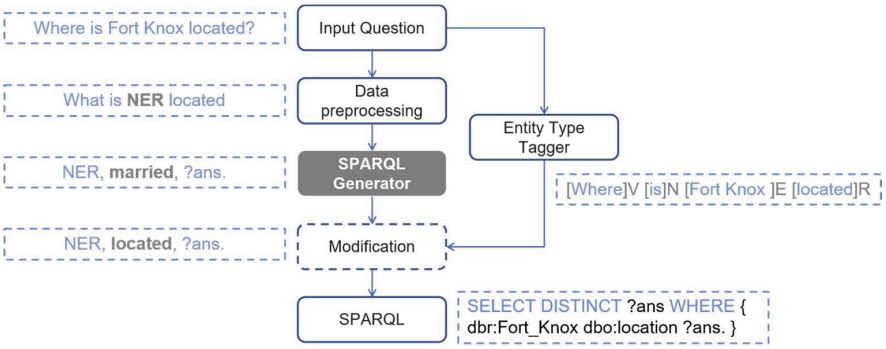

Figure 5 Process of Inference Phase. The process of the inference phase of our presented approach with an example.

turned to 'What is NER located'. Then, the trained SPARQL generator will generate the corresponding SPARQL query for this sentence. However, since the dictionary of our target language is too small to have enough vocabularies for the model to choose, it's highly possible that the model can't find a most suitable property for the sentence. For instance, as shown in Figure 5, the generated SPARQL query by the SPARQL generator is 'NER, married, ?ans.', instead of 'NER, located, ?ans.'. It is obvious that the generated SPARQL query 'NER, married, ?ans.' is a wrong triple. When this situation happens, we employee an entity type tagger to help modify the translation result.

The Modification Phase. The modification component is designed as a remedy for mistranslation problems. Since vocabulary mistranslation is profusely found in translation results, we employ entity type tagger from Chen et al. Study [11] to help modify the result. Thus, the final SPARQL queries are decided jointly by both the result of Transformer model and the entity type tags. V, N, E, R, and C are five main entity type tags, where V represents the interrogatives, N represents the unimportant or stop words, as well as E, R, and C, respectively represent the name entities, relation (property) entities, and classes in the DBpedia. For input question 'Where is Fort_Knox located?', the entity type tagger will give an output 'V-B N E-B R-B'.

For the same question 'Where is Fort Knox located?', the correct translation result should be 'NER, located, ?ans.'. If the question was incorrectly translated into 'NER, married, ?ans.', where the word 'married' does not exist in the input question, it is identified as a vocabulary mistranslated word. Once a vocabulary mistranslated word is found, the mistranslated

words will be replaced by words with E, R, or C tags, depending on their position. Because 'married' is at the relation position, and because 'located' is tagged as R (which means relation), 'married' will be replaced by 'located'. Finally, after entity mapping, we will get a correct SPARQL query 'SELECT DISTINCT ?ans WHERE {dbr:Fort_Knox dbo:location ?ans.}', which is the output sequence of our proposed model. In consideration of space, entity mapping will not be discussed in this paper. For readers who are interested in entity mapping, they can reference Chen et al. Study [11].

4 Experiments

4.1 Dataset and Evaluation Metrix

As one of the main purpose of this paper is to present an approach to automatically generate SPARQL queries and make sure that these queries can be used in an open-domain question answering system, the Large-Scale Complex Question Answering Dataset (LC-QuAD) [9], a commonly used question answering dataset with significantly greater size, variety and complexity, is utilized to evaluate our presented approach, having totally 5000 question-query pairs in it.

In addition to LC-QuAD, datasets from the 7th and the 8th Open Challenge on Question Answering over Linked Data (QALD-7 and QALD-8) [22, 23] are also utilized. This challenge is one of the most popular question answering competition that provides up-to-date benchmark and datasets annually. In the current state of our research, we select only 'simple question' in both QALD-7 and QALD-8 dataset. The 'simple question' here means that in these questions, the corresponding SPARQL queries do not contain any 'filter' in it, such as the SPARQL syntax 'FILTER' or 'ORDER BY', which commonly appear in comparative or superlative questions; however, these 'simple' SPARQL queries are still relatively more complicated than LC-QuAD, where SAPRQL queries in LC-QuAD dataset involve at most three RDF triples, while in the QALD series, there are SPARQL queries with more than five RDF triples and simultaneously having more variables. In QALD-7, there are 186 question-query pairs in training set and 23 pairs in testing set, while in QALD-8, 174 and 34 question-query pairs are included in the training and testing set, respectively.

A combination of evaluation metrics is chosen to present and evaluate the output result in this paper, including the BLEU score as well as accuracy. BLEU score is a quick, inexpensive, and language-independent method of

machine translation evaluation, considering the similarity between the predicted sequence and the gold-standard target sequence [24]. However, as shown in Equation (1) [24], BLEU score somehow lacks consideration of the order and position of the whole generated sequence [2], because the former part (BP; Brevity Penalty) of the equation considers only the lengths of the predicted sequence (i.e. *c*) and the gold-standard target sequence (i.e. *r*) (see Equation (2)), and the latter part utilizes n-gram to compare each word and phrase between the predicted sequence and the gold-standard target sequence. Therefore, accuracy is also employed to evaluate whether every single element in the output sequence is in the right position. We utilize the sklearn.metrics.accuracy_score, which compares token by token between the predicted and the gold-standard target sequence, indicating the percentage of correctly generated tokens.

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} W_n \log P_n\right) \tag{1}$$

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases} \tag{2}$$

4.2 Hyperparameter Experiments and Tuning

Generating SPARQL queries with an NMT model is different from an ordinary machine translation task between two natural languages, since its target language is a query language (SPARQL), where the regularity is higher than any natural language. Moreover, datasets involve natural language question and SPARQL query pairs are usually smaller than machine translation datasets. For instance, as one of the established benchmark for machine translation, WMT 2014 Dataset [25] contains 4.5 million sentence pairs in English-German translation task and 36 million sentence pairs in English-French translation task. On the contrary, the biggest dataset we use (LC-QuAD) contains only 5000 question-query pairs, while there are only hundreds of question-query pairs in both QALD-7 and QALD-8 datasets. Hence, a set of experiments are done to find out the most suitable hyperparameters for SPARQL generation with an NMT model among the three datasets we use.

To start with, we focus on the number of layers in the encoder and decoder, keeping all the other hyperparameters the same as the original Transformer model (see Table 1) except for the embedding size, since a pre-trained

Table 2 BLEU scores (after modification phase) among different layer number

Layer Number	2	4	6
QALD-7	**72.56%**	62.33%	63.86%
QALD-8	**85.22%**	79.81%	82.14%
LC-QuAD	**73.44%**	68.75%	65.25%

Table 3 BLEU scores (after modification phase) among different multi-head number

Multi-head Number	4	8	16
QALD-7	**80.83%**	72.56%	76.18%
QALD-8	**86.84%**	85.22%	85.09%
LC-QuAD	**76.21%**	73.44%	69.45%

Word2Vec embedding is employed in our approach and the dimension of the pre-trained embedding is fixed. As can be observed in Table 2, generally, the BLEU scores sharply decrease when using a larger layer number, meaning that a deep model is unnecessary in this task as the depth of a neural network usually depends on the size of the training dataset.

The layer number is now fixed at 2 in both the encoder and decoder according to the previous experiments. The second step aims to figure out an appropriate multi-head number for our task. In the Transformer model, the embedding of every single word will be projected into many subspaces (heads) during the calculation of attention mechanism to better retrieve information from different dimension, so called the multi-head attention [17]. As shown in Table 3, increasing the multi-head number does not seem to be more effective in our task, while using 4 as the multi-head number of our model is already good enough due to the embedding size as well as the lower complexity and the fewer vocabulary of our target language.

As learning rate is considered as one of the most important hyperparameter while training neural networks, two different strategies on tuning learning rate are conducted in this step.

For the first one, we follow the original Transformer model using the Noam learning rate scheduler [17]. In this scheduler, learning rate starts from a small value, linearly increases during the warm up steps to stabilize the neural network, and then exponentially decays over steps (see Equation (3)). Based on our various experiments, the batch size and the number of epochs were fixed at 32 and 150; respectively. Although the default warm up step number is 4000 for the original Transformer model, only 900 steps (150 epochs * 6 batches) are required while training the QALD-7 and QALD-8 datasets. Therefore, smaller warm up step numbers for QALD-7 and QALD-8

Table 4 BLEU scores (after modification phase) among different warm up step of the Noam learning rate scheduler on QALD-7 and QALD-8

Warm Up Steps	100	200	300	4000
QALD-7	81.35%	75.29%	**83.13%**	80.83%
QALD-8	**88.83%**	79.81%	87.73%	86.84%

Table 5 BLEU scores (after modification phase) among different warm up step of the Noam learning rate scheduler on LC-QuAD

Warm Up Step	500	1000	2000	4000	4500
LC-QuAD	68.94%	72.72%	73.50%	**76.21%**	74.49%

Table 6 BLEU scores (after modification phase) among different learning rate

Learning Rate	Noam Scheduler	0.0001	0.00015	0.0002	0.0005	0.001
QALD-7	83.13%	80.83%	**83.49%**	78.36%	73.74%	79.80%
QALD-8	88.83%	86.19%	**90.13%**	84.83%	83.37%	82.31%
LC-QuAD	76.21%	74.15%	**76.32%**	74.54%	75.42%	71.59%

were experimented, and both 100 and 300 were found to outperform the default settings, as shown in Table 4. As for the LC-QuAD dataset (see Table 5), due to its larger size, 4000 seems to be the best warm up step number among its 18750 training steps (150 epochs * 125 batches).

$$\mathrm{lrate} = \mathrm{d}_{\mathrm{model}}^{-0.5} \cdot \min(\mathrm{step\_num}^{-0.5}, \mathrm{step\_num} \cdot \mathrm{warmup\_steps}^{-1.5}) \quad (3)$$

However, learning rate scheduler and warm up step strategy are more suited for very deep neural networks with dozens or even hundreds of layers [26]. Generally, they are used in convolutional neural network (CNN) for image processing but seldom employed in models for natural language processing tasks which usually have less layer number [26]. Therefore, we also try to fix learning rate values rather than using a learning rate scheduler. As shown in Table 6, by fixing the learning rate on 0.00015, all the three datasets can reach the highest BLEU score.

4.3 Results and Discussion

The final evaluation results of our approach are shown in Table 7, including the effect of the modification phase we designed in this study. The BLEU score of QALD-8 reached an outstanding 90.13% after the modification phase, followed by 83.49% and 76.32% on QALD-7 and LC-QuAD, respectively. The accuracies of QALD-7, QALD-8, and LC-QuAD are 76.69%,

Table 7 Results before and after the Modification Phase

Dataset	Evaluation Metrix	SPARQL Generator	Entity Type Tags Modified
QALD-7	BLEU Score	60.95%	83.49% (+22.54%)
	Accuracy	60.74%	76.69% (+15.95%)
QALD-8	BLEU Score	58.45%	90.13% (+31.68%)
	Accuracy	49.09%	88.15% (+39.06%)
LC-QuAD	BLEU Score	75.06%	76.32% (+1.26%)
	Accuracy	71.27%	72.00% (+0.73%)

Table 8 Comparison among our approach and the other studies for SPARQL generation on LC-QuAD dataset

Evaluation	Our Approach		Yin	Diomedi and
Metrix	SPARQL Generator	Entity Type Tags Modified	et al. [2]	Hogan [3]
BLEU Score	75.06%	76.32%	59.54%	59.30%
Accuracy	71.27%	72.00%	8.00%	14.00%

88.15%, 72.00%; respectively. It is noted that, the translation results of our SPARQL generator without the modification phase are also well performed no matter in BLUE score or accuracy among open-domain question answering datasets.

A comparison on BLEU scores and accuracies among our presented approach and the previous ones [2, 3] is summarized in Table 8. It is obvious that our approach, with or without modification phase, is significantly superior than all previous studies. It is noted that the best performances of the two previous studies are based on ConvS2S model. Lc-QuAD seems to have enough vocabularies to overcome most mistranslation problems, so the translation result without the modification phase is already good and way better than the previous studies [2, 3]. Yin et al. [2] performed badly on accuracy, but still get acceptable BLEU score. The low accuracy was caused by many misplaced punctuations and tokens in the generated sequence. Diomedi and Hogan [3] attempted to overcome the problem by translating the natural language questions to templates, however, their performance is still limited.

5 Conclusion

In this work, we propose an NMT-based approach using the Transformer model to automatically translate natural language questions to SPARQL queries. Our approach is evaluated with QALD-7, QALD-8, and LC-QuAD

datasets, reaching an outstanding result among open-domain question answering datasets. Considering our high accuracy, this SPARQL generation approach is feasible to develop as a complete and well performed end-to-end question answering system.

In addition, there is a main drawback of our current approach that can be improved in the future. The current approach heavily relies on named entity recognition and entity type tagging, which can be a waste of training cost. Moreover, if the effectiveness of the named entity recognizer and the entity type tagger is not satisfactory, the final performance of our approach will reduce greatly. For future works, generating a pre-trained embedding under DBpedia entities can be an option to replace the modification phase.

References

[1] Prud'hommeaux, E., and A. Seaborne. 2008. SPARQL Query Language for RDF. Available: https://www.w3.org/TR/rdf-sparql-query/.

[2] Yin, X., D. Gromann, and S. Rudolph. 2021. Neural Machine Translating from Natural Language to SPARQL. *Future Generation Computer Systems*. pp. 510–519.

[3] Diomedi, D., and A. Hogan. 2021. Question Answering over Knowledge Graphs with Neural Machine Translation and Entity Linking. arXiv:2107.02865 [cs.AI].

[4] Bizer, C., J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. 2009. DBpedia - A crystallization point for the Web of Data. *Web semantics*. 7(3): pp. 154–165.

[5] Sutskever, I., O. Vinyals, and Q. Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*. Montreal, Canada.

[6] Soru, T., E. Marx, D. Moussallem, G. Publio, A. Valdestilhas, D. Esteves, and C. B. Neto. 2017. SPARQL as a Foreign Language. *SEMANTiCS CEUR Workshop Proceedings 2044*. Amsterdam, The Netherlands.

[7] Soru, T., E. Marx, A. Valdestilhas, D. Esteves, D. Moussallem, and G. Publio. 2018. Neural Machine Translation for Query Construction and Composition. *ICML workshop on Neural Abstract Machines & Program Induction v2*. Stockholm, Sweden.

[8] Hartmann, A.-K., T. Soru, and E. Marx. 2018. Generating a Large Dataset for Neural Question Answering over the DBpedia Knowledge

Base. *Workshop on Linked Data Management and WEBBR*. Vienna, Austria.

[9] Trivedi, P., G. Maheshwari, M. Dubey, and J. Lehmann. 2017. LC-QuAD: A Corpus for Complex Question Answering over Knowledge Graphs. *Lecture Notes in Computer Science*. Springer, Cham. pp. 210–218.

[10] Diefenbach, D., A. Both, K. Singh, and P. Maret. 2020. Towards a Question Answering System over the Semantic Web. *Semantic Web.* 11(3): pp. 421–439.

[11] Chen, Y.-H, Lu, E. J.-L., and Ou, T.-A., 2021. Intelligent SPARQL Query Generation for Natural Language Processing Systems. *IEEE Access*. 9: pp. 158638–158650.

[12] Lu, E. J.-L., and C.-H. Cheng. 2020. Multiple Classifiers for Entity Type Recognition. *Conference on Smart Computing.* Penghu, Taiwan.

[13] Kuo, C.-Y., and E. J.-L. Lu. 2021. A BiLSTM-CRF Entity Type Tagger for Question. *IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology*. Indonesia.

[14] Xu, K., S. Zhang, Y. Feng, and D. Zhao. 2014. Answering natural language questions via phrasal semantic parsing. *Natural Language Processing and Chinese Computing*. Berlin, Heidelberg.

[15] Hu, S., L. Zou, Y. J. Xu, H. Wang, and D. Zhao. 2018. Answering Natural Language Questions by Subgraph Matching over Knowledge Graphs. *IEEE Transactions on Knowledge and Data Engineering*. 30(5): pp. 824–837.

[16] Gehring, J., M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. 2016. Convolutional Sequence to Sequence Learning. *34th International Conference on Machine Learning*. Sydney, Australia.

[17] Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. 2017. Attention Is All You Need. *31st Conference on Neural Information Processing Systems.* Long Beach, CA, USA.

[18] Lin, J.-H., and E. J.-L. Lu. 2021. An NMT-based Approach to Translate Natural Language Questions to SPARQL Queries. *International Conference on IT Convergence and Security*. Virtual Conference.

[19] Liang, S., K. Stockinger, T. Mendes de Farias, M. Anisimova, and M. Gil. 2021. Querying knowledge graphs in natural language. *Journal of Big Data*. 8(1): pp. 1–23.

[20] Firat, O., K. Cho, B. Sankaran, F. T. Yarman Vural, and Y. Bengio. 2017. Multi-way, multilingual neural machine translation. *Computer speech & language.* 9: pp. 236–252.

[21] Wu, Y., M. Schuster, Z. Chen, Q. V. Le, and M. Norouzi. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv:1609.08144 [cs.CL].

[22] Usbeck, R., A.-C. N. Ngomo, B. Haarmann, A. Krithara, M. Röder, and G. Napolitano. 2017. 7th Open Challenge on Question Answering over Linked Data (QALD-7). *SemWebEval 2017: Semantic Web Challenges*. Springer, Cham.

[23] Usbeck, R., A.-C. N. Ngomo, F. Conrads, M. Röder, and G. Napolitano. 2018. 8th challenge on question answering over linked data (QALD-8). *Language.* 7(1): pp. 51–57.

[24] Papineni, K., S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics.* Philadelphia, Pennsylvania, USA.

[25] Bojar, O., C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, A. Tamchyna. 2014. Findings of the 2014 Workshop on Statistical Machine Translation. *Proceedings of the Ninth Workshop on Statistical Machine Translation.* Maryland, USA.

[26] Wen, L., X. Li, and L. Gao. 2021. A New Reinforcement Learning Based Learning Rate Scheduler for Convolutional Neural Network in Fault Classification. *IEEE Transactions on Industrial Electronics.* 68(12): pp. 12890–12900.

Biographies

Jia-Huei Lin received the bachelor's degree in computer science and information engineering from National Taiwan Normal University in 2019. She is currently studying for her master's degree majoring in Management Information in National Chung Hsing University. She has engaged in researches about natural language processing, question answering system, and neural machine translation.

Eric Jui-Lin Lu received the B.A. degree from the National Chiao-Tung University, Tsin-Chu in 1982. Later on, he received his MSBA degree from San Francisco State University, San Francisco in 1990. He received his Ph.D. degree in computer science from Missouri University of Science and Technology (formerly University of Missouri-Rolla), Missouri in 1996. He is currently a professor with the Department of Management Information Systems, National Chung Hsing University. His research interests include machine learning, natural language question answering, and semantic web.