

# A system for translating natural language questions into SPARQL queries with neural networks: Preliminary results

(Discussion Paper)

Manuel Alejandro Borroto<sup>1</sup>, Francesco Ricca<sup>1</sup> and Bernardo Cuteri<sup>1</sup>

<sup>1</sup>University of Calabria, Via Pietro Bucci, Rende, Cosenza, 87036, Italy

## Abstract

The development of knowledge bases has gathered nowadays large volumes of information concerning multiple domains. Unfortunately, access to this information is complicated for those users unfamiliar with the SPARQL query language and the knowledge base definition. In this paper, we present preliminary results on a system for automatic translation of natural language questions into SPARQL queries. Our method uses Neural Machine Translation and Named Entity Recognition tasks that complement each other to obtain a final query ready to be executed. We demonstrate the potential of our approach by presenting its results on the Monument dataset, which is a recently released dataset for Question Answering on the well-known DBpedia knowledge base.

## Keywords

Knowledge base, Question Answering, Neural network

## 1. Introduction

Today we live in what is known as the Digital Age. How knowledge is generated and shared has changed dramatically, digital formats and the internet have made information much more accessible than the old non-virtual format. As evidence of this, we now have vast and complex knowledge bases which allow gathering large volumes of information through the intercommunication of thousands of datasets referring to various domains in what is known as Linked Data. This means that the people have access to a large amount of information never thought, and the DBpedia [1] project is a real example of that, which is one of the most popular knowledge bases nowadays.

The problem is that the search and retrieval of the information stored in this way can be a hard task for lay users because it is necessary to know the structure of the knowledge base and the appropriate query languages, such as SPARQL [2]. As a result, natural language Question Answering (QA) has taken a central role in the area of the Semantic Web to address such issues. A group of QA approaches, especially the most recent, have begun to take advantage of the

---


*SEBD 2021 - The 29th Italian Symposium on Advanced Database Systems, September 5-9, 2021, Pizzo Calabro (VV), Italy*

✉ manuel.borroto@unical.it (M. A. Borroto); francesco.ricca@unical.it (F. Ricca); bernardo.cuteri@unical.it (B. Cuteri)

🆔 0000-0001-8218-3178 (F. Ricca); 0000-0001-5164-9123 (B. Cuteri)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

great development achieved by Deep Learning and started to use deep neural networks to tackle the problem, proposing systems for the automatic translation from natural language questions to SPARQL queries, removing all technical complexity to the final users.

In this context, we propose a system for the automatic translation of natural language questions into SPARQL queries. More specifically we employ LSTM [3] neural networks due to the proven effectiveness for Natural Language Processing that they have demonstrated. The system consists of two parts. The first one translates the questions in natural language into a SPARQL template using an LSTM encoder-decoder model, which is the state-of-the-art for these types of tasks [4]. Whereas the second part is a model for Named Entity Recognition [5], also based on LSTM networks, and responsible for extracting the entities from the question to finally combine the results and create a SPARQL query ready to be executed. Besides, we introduce a formal definition of a dataset format that greatly reduces the output space and is essential for the proper functioning of the system and also allows us to tackle the problem with the out-of-vocabulary (OOV) words of the training set, a major weakness of the majority of the related approaches today.

We demonstrate the potential of our approach by presenting its results on the Monument dataset, which is a recently released dataset for Question Answering on the well-known DBpedia knowledge base.

The remainder of the paper is structured as follows. Section 2, we talk about related works. In section 3, we go into the particular details of our approach. Section 4 focuses on the discussion of experiments and results. Finally, we provide some conclusions and aspects for future work.

## 2. Related Work

**Pattern-based.** The idea of employing query patterns for mapping questions to SPARQL-queries was already exploited in the literature [6, 7]. The approach presented by Pradel and Ollivier [6] also adopts named entity recognition but applies a set of predefined rules to obtain all the query elements and their relationships. The approach by Steinmetz et. al [7] has 4 phases, firstly, the question is parsed and the main focus is extracted, then general queries are generated from the phrases in natural language according to predefined patterns, and finally, makes a subject-predicate-object mapping of the general question to triples in RDF. Despite both of the above-mentioned approaches performed well in selected benchmarks, they rely on patterns and rules defined manually for all existing types of questions. A limit that is not present in our proposal.

**Deep Learning-based.** In the Seq2SQL approach [8] an LSTM Seq2Seq model is used to translate from natural language to SQL queries. The interesting thing about this approach is that they use *Reinforcement Learning* to guide the learning. The usage Encoder-Decoder model based in LSTM with an attention mechanism to associate a vocabulary mapping between natural language and SPARQL also was proposed in the literature [9] obtaining good results.

The *Neural SPARQL Machines (NSpM)* [10] approach is based on the idea of modifying the SPARQL queries to treat them as a foreign language. To achieve this, they encoded the brackets, URIs, operators, and other symbols, making the tokenization process easier. The resulting

dataset was introduced in a Seq2Seq model responsible for performing the question-query mapping. The same authors created the *DBNQA dataset* [11], and their model was tested on a subdomain referring to monuments and evaluated using the purely syntactic BLEU score [10]. As a consequence, it performs well in reproducing the syntax of the gold query but is less able to generalize to unseen natural language questions and OOV words when compared with our approach.

The query building approach by Chen et al. [12] features two stages. The first stage consists of predicting the query structure of the question and leverages the structure to constrain the generation of the candidate queries. The second stage performs a candidate query rank. As in our approach, Chen et al. [12] uses BiLSTM networks, but query representation is based on abstract query graphs.

Also, we report that eight different models based on RNNs and CNNs were compared by Yin and colleagues [13]. In this large experiment, the ConvS2S [14] model proved to be the best.

For completeness, we studied another related line of work that aims to translate the natural language questions into SQL queries. The work proposed by Yu et. al [15] introduces a large-scale, complex, and cross-domain semantic parsing and text-to-SQL dataset. To validate the work contribution, they used the proposed dataset to train different models to convert text to SQL queries. Most of the models were based on a Seq2Seq architecture with attention, demonstrating an adequate performance. Another interesting case of study is the editing-based approach for text-to-SQL generation introduced by Zhang et. al [16]. They implement a Seq2Seq model with Luong’s attention, using BiLSTMs and BERT embeddings. The approach demonstrates to perform well on SPaC and Spider datasets, outperforming the related work in some cases.

Our architecture addresses the issues connected with the translation resorting to specific tools, an aspect that is not present in mentioned works. Moreover, existing approaches based on NMT do nothing special to deal with OOV words.

### 3. Translating Natural Language Questions to SPARQL

Knowledge bases (KB) are a rich source of information related to a great variety of domains, which can be accessed by experts of formal query languages. The potential of exploiting knowledge bases can be greatly increased by allowing any user to query the ontology by posing questions in natural language.

In this paper, this problem is seen as the following Natural Language Processing task: Given an RDF knowledge base  $O$  and a question  $Q_{nat}$  in natural language (to be answered using  $O$ ), translate  $Q$  into a SPARQL query  $S_{Q_{nat}}$  such that the answer to  $Q_{nat}$  can be obtained by running  $S_{Q_{nat}}$  on the underlying ontology  $O$ .

The starting point is a training set containing a number of pairs  $\langle Q_{nat}, G_{Q_{nat}} \rangle$ , where  $Q_{nat}$  is a natural language question, and  $G_{Q_{nat}}$  is a SPARQL query, called the *gold query*. The gold query is a SPARQL query that models (i.e., allows to retrieve from  $O$ ) the answers to  $Q_{nat}$ . The training set has to be used to learn how to answer questions posed in natural language using  $O$ , so that, given a question in natural language  $Q_{nat}$ , the QA system can generate a query  $S'_{Q_{nat}}$  that is equivalent to the gold query  $G_{Q_{nat}}$  for  $Q_{nat}$ , i.e., such that

**Table 1**

$\langle \text{Question}, \text{Query} \rangle$  pair for *Who painted the Mona Lisa?*

Question	Query
Who painted the Mona Lisa?	select ?a where {dbr:Mona_Lisa dbo:author ?a.}

$answers(S'_{Q_{nat}}) = answers(G_{Q_{nat}})$ .<sup>1</sup> In particular, we approach this problem as a machine translation task, that is, we compute  $S'_{Q_{nat}}$  as  $S'_{Q_{nat}} = Translate(Q_{nat})$ , where *Translate* is the translation function implemented by our QA System, called *sparql-qa*.

In the remainder, we first present an intermediate format conceived to boost the training time of the entire process and reduce the impact of words that reference individuals that are not mentioned in the training set, in turn, we describe our translation modules that take as input the dataset in the new format.

### 3.1. A new data set format

In general, NL to SPARQL datasets are composed of a set of pairs  $\langle Q_{nat}, G_{Q_{nat}} \rangle$ . In such a common type of representation, the named entities found in the question are typically represented directly by their URIs in the SPARQL query, but this transformation is hard to learn from mere examples, and the trained system would fail if the transformation can not be described as simple rules. This is an issue, especially in large ontologies, where there is a huge number of resources.

A dataset in QQT is composed of a set of triples in the form  $\langle \text{Question}, \text{QueryTemplate}, \text{Tagging} \rangle$ , where *Question* is a natural language question, and *Tagging* marks which parts of *Question* are entities, and *QueryTemplate* is a SPARQL query template with the following modifications: (i) The KB resources are replaced by one or more variables; (ii) A new triple is added for each variable in the form "*?var rdfs:label placeholder*". *Placeholders* are meant to be replaced by substrings of *Question* depending on *Tagging*.

In Table 1 we show an example of a  $\langle Q_{nat}, Q_{sparql} \rangle$  pair for the question *Who painted the Mona Lisa?*, while Table 2 shows the corresponding  $\langle \text{Question}, \text{QueryTemplate}, \text{Tagging} \rangle$  triple in the QQT format.

In table 2 the term  $\$1$  denotes a placeholder, where 1 means that it has to be replaced by the first entity occurring in the question, that is *Mona Lisa* as represented by *B* and *I* in *Tagging*. Note that, in the QQT format, the query template does not contain any DBpedia resource, thus the learning model (which is the neural network in our case) does not need to understand that

<sup>1</sup>Note that we are interested in computing the answers and not in syntactically reproducing the gold query.

**Table 2**

$\langle \text{Question}, \text{QueryTemplate}, \text{Tagging} \rangle$  triple for *Who painted the Mona Lisa?*

Question	QueryTemplate	Tagging
Who painted the Mona Lisa?	select ?a where { ?w dbo:author ?a. ?w rdfs:label \$1 }	O O O B I O

Mona Lisa stands for the *dbp:Mona\_Lisa* resource and the *QueryTemplate* is exactly the same for all questions asking the author of a given artwork.

### 3.2. The translation modules

Our approach consists of two deep neural networks, the first one specialized in Neural Machine Translation (NMT) based on the well-known Seq2Seq [4] model and the second one used for extracting the entities from the question using the Named Entity Recognition (NER) technique.

#### 3.2.1. Neural Machine Translation

The network focused on NMT is used to translate the question into a SPARQL *QueryTemplate*. The network is based on an Encoder-Decoder model with *Luong's attention* [17], in which the Encoder extracts semantic content from the question in natural language and encodes it into a fixed-dimensional vector representation  $V$ . Instead, the Decoder tries to decode  $V$  into a sequence in the output language (*QueryTemplate*).

The Encoder is composed of an input layer that receives a question in natural language converted into a sequence of word-embeddings obtained by mean of FastText [18], in the form  $\{x_1, x_2, \dots, x_t\}$ , where  $x_t$  is the vector representation of the word  $t$  in the sentence. Next, we use a Bidirectional LSTM (BiLSTM) to summarize  $\{x_1, x_2, \dots, x_t\}$  into  $V$ , in forward and reverse orders.  $V$  is formed by concatenating the last hidden states in the two directions.

On the other hand, during the training process, the Decoder is responsible for calculating the word-embeddings of the output language tokens (SPARQL), which is used together with the vector  $V$ , provided by the Encoder, as input to a Luong-Decoder layer. This layer is responsible for decoding the sentence supported by the attention mechanism. Finally, the values are feed to a Fully Connected Network with a Softmax activation function that predicts the output sequence by calculating the conditional probability over the output vocabulary. Figure 1 shows the described network architecture.

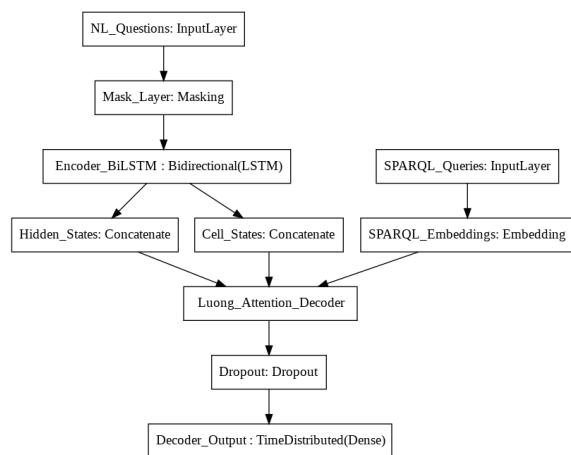


Figure 1: NMT neural network architecture

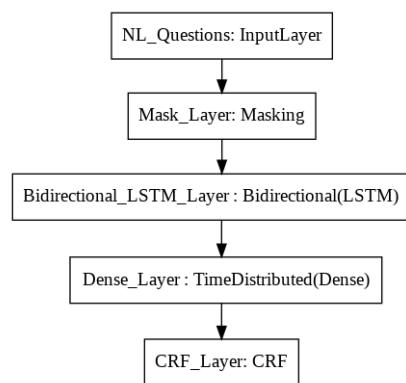


Figure 2: NER neural network architecture

### 3.2.2. Named Entity Recognition

To perform the entity recognition, we created a BiLSTM-CRF [5] network that constitutes state-of-the-art for this type of task. In this case, we again used FastText to obtain the word-embeddings and deal with OOV words. The model is composed of an input layer that receives the sequences of embeddings, followed by a BiLSTM connected to a Fully Connected layer. Finally, the information flows through a CRF layer that predicts the final sequence of tags. Figure 2 shows the described network architecture.

Finally, we mixed the results of both networks to obtain the final query  $S'_{Q_{nat}}$ . Here, the placeholders in the *QueryTemplate* are replaced by the corresponding entities obtained with the NER network.

To better understand how *sparql-qa* works, we can translate, for example, the question: *Where is Washington Monument located?* First, the question is cleaned and split into tokens and then converted into a sequence of fastText word-embeddings. Then, the sequence is processed by the two networks used by our system. The NMT network translates the question into the corresponding *QueryTemplate*:

```
SELECT DISTINCT ?a WHERE { ?w dbo:location ?a . ?w rdfs:label $1 }
```

successively the NER network calculates the tagging sequence: *O O B I O*, where it is indicated that the entity to be considered is *Washington Monument* (Positions 2 and 3 of the tagging sequence, respectively). Finally, in the composition phase, the results are mixed, obtaining the final query:

```
SELECT DISTINCT ?a WHERE { ?w dbo:location ?a .  
                           ?w rdfs:label "Washington Monument"@en }
```

It is important to note that the previous example describes the translation operation in general terms and does not go into the more complicated details of the process.

## 4. Experiments on Monument dataset

**Setup.** The Monument dataset was proposed as part of the Neural SPARQL Machines (NSpM) [10] research. It contains 14,778 question-query pairs about the instances of type monument present in DBpedia.

We compared our system with the state-of-the-art, thus, we have trained the Learner Module of NSpM as it was done in [10], where the authors proposed two instances of the Monument dataset that we will denote by Monumet300 and Monument600 containing 8,544 and 14,788 pairs, respectively. In both cases, the dataset split fixes 100 pairs for both validation and test set and keeps the rest for the training set. All the data is publicly available in the NSpM GitHub project.<sup>2</sup>

We have implemented our system, called *sparql-qa*, by using Keras, a well-known framework for machine learning, on top of TensorFlow. We trained the networks by using Google Colaboratory, which is a virtual machine environment hosted in the cloud and based on Jupyter

---

<sup>2</sup><https://github.com/LiberAI/NSpM/tree/master/data>

**Table 3**  
Comparison on Monument datasets.

	Mon300			Mon600		
	P	R	F1	P	R	F1
<b>NSpM</b>	0.860	0.861	0.852	0.929	0.945	0.932
<b><i>sparql-qa</i></b>	0.78	0.78	0.78	0.791	0.791	0.791

Notebooks. The environment provides 12GB of RAM and connects to Google Drive. To train our system, we first performed hyperparameter tuning focused on three metrics: embedding-size of the target language, batch size, and LSTM hidden units. The task was performed by using a grid search method. We set the number of epochs to 5, shuffling the dataset at the end of each one. After tuning, we set the hyperparameters of the two networks as follows: embedding-size is set to 300, LSTM hidden units are set to 96, and batch size is set to 64.

For comparing performance, we adopted the macro precision, recall, and F1-score measures, which are the most used ones to assess this kind of system.

**Results.** Results of the execution reported in Table 3 show that *sparql-qa* performs reasonably well, reaching F1-score values greater than 0.7. On the other hand, NSpM achieves better results.

We have investigated why our system could not provide an optimal answer for some questions. This analysis evidenced that the performance of our approach is mainly affected by problems in the dataset. Indeed, there is a set of questions that lacks context to determine specific expected URIs. For example, for the question “*What is Washington Monument related to?*” our system uses “Washington Monument”, but the gold query uses the specific URI: *Washington\_Monument\_(Baltimore)*. Note that there is no reference to Baltimore in the question text, and there are Washington Monuments also in Milwaukee and Philadelphia, according to DBpedia. Surprisingly, the compared system can often use the specific URI of the gold query even without context. Thus, we run another experiment to better outline the issue. We create a new test set of 200 pairs by using the templates provided by NSpM and a randomly selected set of unseen monument entities extracted from DBpedia. Table 4 shows that our approach has the same good performance (F1 score greater than 0.78) and performs much better than NSpM that is not able to generalize to deal with OOV (F1 of 0.11).

Another cause that affects our approach is the correctness with which the named entities are written. Sometimes the entities mentioned in the question do not match the *rdfs:label* property value, and sometimes they are referenced using acronyms. In these cases, our system will not give the expected answers because it cannot reference the right DBpedia resources. To address

**Table 4**  
Comparison with OOV entities on Monument datasets.

	Mon300			Mon600		
	P	R	F1	P	R	F1
<b>NSpM</b>	0.097	0.123	0.101	0.11	0.11	0.11
<b><i>sparql-qa</i></b>	0.795	0.795	0.795	0.785	0.785	0.785



these issues, we plan to use *Named Entity Linking (NEL)* [19], which allows us to determine accurately which DBpedia resources are present in the question.

Finally, for completeness, we report that the intermediate format allows us to save 40% of training time.

## 5. Conclusions and Future Work

The paper presents preliminary results on an approach for querying SPARQL knowledge bases by using natural language. We combine in our system both neural machine translation and named entity recognition modules and focus on attenuating the impact of the OOV words, an important issue that is not well considered in existing approaches. Our system showed good preliminary results on the Monument dataset and demonstrated a more general and robust behavior than state-of-the-art approaches.

In future work, we plan to extend our system to improve translation performance by integrating other NLP tools, such as Named Entity Linking and BERT contextual word embeddings. We also plan to extend our experiments by considering other well-known QA benchmarks.

## Acknowledgments

This work was partially supported by the Italian Ministry of Economic Development (MISE) under projects “MAP4ID - Multipurpose Analytics Platform 4 Industrial Data”, N. F/190138/01-03/X44.

## References

- [1] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, et al., Dbpedia—a large-scale, multilingual knowledge base extracted from wikipedia, *Semantic Web 6* (2015) 167–195.
- [2] W3C, Semantic web standards, 2014. URL: <https://www.w3.org>.
- [3] S. Hochreiter, J. Schmidhuber, Long short-term memory, *Neural Computation* 9 (1997) 1735–1780.
- [4] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: *NIPS*, 2014, pp. 3104–3112.
- [5] Z. Huang, W. Xu, K. Yu, Bidirectional LSTM-CRF models for sequence tagging, *CoRR abs/1508.01991* (2015).
- [6] C. Pradel, O. Haemmerlé, N. Hernandez, Natural language query interpretation into sparql using patterns, 2013.
- [7] N. Steinmetz, A. Arning, K. Sattler, From natural language questions to SPARQL queries: A pattern-based approach, in: *BTW*, volume P-289 of *LNI*, Gesellschaft für Informatik, Bonn, 2019, pp. 289–308.
- [8] V. Zhong, C. Xiong, R. Socher, Seq2sql: Generating structured queries from natural language using reinforcement learning, *CoRR abs/1709.00103* (2017).



- [9] F. F. Luz, M. Finger, Semantic parsing natural language into SPARQL: improving target language representation with neural attention, CoRR abs/1803.04329 (2018).
- [10] T. Soru, E. Marx, D. Moussallem, G. Publio, A. Valdestilhas, D. Esteves, C. B. Neto, SPARQL as a foreign language, SEMANTiCS 2017 - Posters and Demos (2017). URL: <https://arxiv.org/abs/1708.07624>.
- [11] A. Hartmann, E. Marx, T. Soru, Generating a large dataset for neural question answering over the DBpedia knowledge base (2018).
- [12] Y. Chen, H. Li, Y. Hua, G. Qi, Formal query building with query structure prediction for complex question answering over knowledge base, in: IJCAI, 2020.
- [13] X. Yin, D. Gromann, S. Rudolph, Neural machine translating from natural language to SPARQL, CoRR abs/1906.09302 (2019).
- [14] J. Gehring, M. Auli, D. Grangier, D. Yarats, Y. N. Dauphin, Convolutional sequence to sequence learning, in: ICML, volume 70 of *Proc. of ML Research*, PMLR, 2017, pp. 1243–1252.
- [15] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, et al., Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task, arXiv preprint arXiv:1809.08887 (2018).
- [16] R. Zhang, T. Yu, H. Y. Er, S. Shim, E. Xue, X. V. Lin, T. Shi, C. Xiong, R. Socher, D. Radev, Editing-based sql query generation for cross-domain context-dependent questions, arXiv preprint arXiv:1909.00786 (2019).
- [17] M. Luong, H. Pham, C. D. Manning, Effective approaches to attention-based neural machine translation, arXiv preprint arXiv:1508.04025 (2015).
- [18] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, *TACL* 5 (2017) 135–146.
- [19] W. Shen, J. Wang, J. Han, Entity linking with a knowledge base: Issues, techniques, and solutions, *IEEE Transactions on Knowledge and Data Engineering* 27 (2014) 443–460.