SPARQL-QA enters the QALD challenge

Manuel Borroto[1], Francesco Ricca[1], Bernardo Cuteri[1], and Vito Barbara[1]

University of Calabria, Rende (CS) 87036, Italy
{manuel.borroto,francesco.ricca,bernardo.cuteri,barbara.vito}@unical.it

Abstract. The large public knowledge bases available today empower the community with an invaluable amount of information covering multiple domains. Unfortunately, accessing these data sources can be difficult without the proper knowledge of the KB structure and a query language, such as SPARQL. Question Answering over Knowledge Bases (KBQA) proposes to solve this problem by allowing users to pose questions in natural language to extract the desired information from the KBs, abstracting them from any technical complexity. In this context, the QALD challenges propose to evaluate and encourage the development of KBQA systems. In this paper, we describe *sparql-qa*, a system for question answering over KBs that have been submitted to the QALD-10 challenge. Our method uses Neural Machine Translation and Named Entity Recognition tasks that complement each other to create SPARQL queries. The experiments were performed using the Wikidata QALD-10 benchmark.

1 Introduction

Today, search and retrieval of the rich information stored in large knowledge bases is still a challenging task for those lay users that do not know the structure of the knowledge base and the appropriate query languages, such as SPARQL. As a result, AI techniques for natural language Question Answering (QA) have taken a central role in the area of the Semantic Web to address such issues. Indeed, systems able to translate questions posed in natural language in SPARQL queries have the potential of overcoming this problem because they can remove all technical complexity to the final users.

In this paper, we describe *sparql-qa* [3], an AI system for QA over knowledge bases. The core of our architecture is based on a Neural Machine Translation (NMT) [1] module, which is based on Bidirectional Recurrent Neural Networks [11], *trained side-by-side* with a Named Entity Recognition (NER) module, implementing a BiLSTM-CRF network [7]. The NMT module translates the input NL question into a SPARQL template, whereas the NER module extracts the entities from the question. The combination of the outputs of the two modules results in a SPARQL query ready to be executed.

In the following, we provide an overview of the system that participated in the 10th Question Answers over Linked Data (QALD)[1] challenge edition.

[1] http://qald.aksw.org/

2 System Architecture

The potential of exploiting knowledge bases can be 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} is obtained by running $S_{Q_{nat}}$ on 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 $answers(S'_{Q_{nat}}) = answers(G_{Q_{nat}})$. Basically, we compare the answers, and we are not interested in reproducing syntactically the gold query. 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 implemented by our QA system, called *sparql-qa*. Most of the solutions proposed up to now to convert from natural language to SPARQL make use of various techniques, either using patterns or deep neural networks.

To reduce the impact of the words out of the vocabulary (WOOV) and improve the training time of the entire process, we introduce in *sparql-qa* some remedies, including a new format to represent an NL to SPARQL dataset. In particular, *sparql-qa* implements a neural-network-based architecture (see Figure 1) for question answering that accomplishes the objective by resorting to a novel combination of tools. The architecture is composed of three main modules: *Input preparation, Translation, and Assembling*, as shown in Figure 1 ($left$). Each module is described in the following.

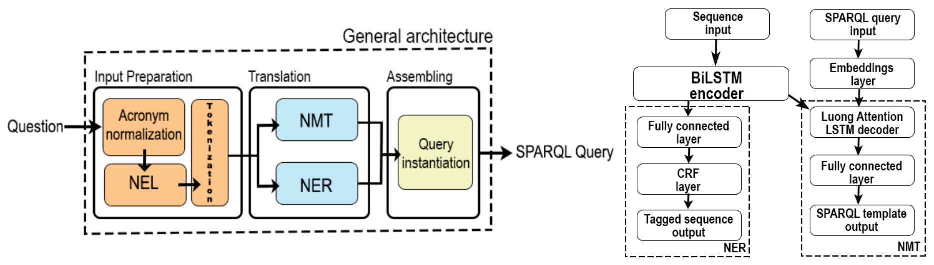

Fig. 1. System architecture (left) and model for joint training of NMT and NER (right).

2.1 Input preparation

In this phase, the input sentence is processed in such a way that it is polished to attenuate linguistic noise (e.g., shifts in spelling, grammar, punctuation, entity identification) and also recast to be used as input for the subsequent phase.

Acronyms normalization. In this step, acronyms are converted to the corresponding names. For example, the UK acronym becomes *United Kingdom*. Acronyms regularly refer to KB resources inside the SPARQL query, and we need to replace them with the full name, which is more similar to the KB resource name. In our approach, this is particularly useful for handling acronyms of countries. To perform this task, we rely on two libraries. The first one is *spaCy* [6], a well-known tool for NLP tasks. This library helps us to identify the acronyms thanks to their powerful NER mechanism. Then we use the *Country Converter* (COCO) library to obtain the original name.

Fixing entities with NEL. We apply this preprocessing step to identify the entities in Q_{nat} and replace them with the label used in the KB because our approach heavily relies on the correct spelling of the entities. For example, in the question: *Where was the president Kennedy born?*, we look to transform the entity *"Kennedy"* to *"John F. Kennedy"* which is the right label used to identify the resource. In our implementation, we face this problem by using *Named Entity Linking* (NEL), also referred to as *Named Entity Disambiguation*, which is the task of linking entities mentioned in the text with their corresponding entities in a target KB [10]. To this end, we employed *DBpedia Spotlight* [4].

Tokenization. The tokenization process is a fundamental step in almost all NLP methods. It is the task of chopping a text up into pieces called *tokens*, which are usually words. During this process, Q_{nat} is cleaned by filtering out undesired characters such as punctuation marks and converted into a sequence of words S. The sequence is converted then into a sequence of word embeddings. To mitigate the dependency on the input vocabulary (English) and reduce the impact of the WOOV, we use the pre-trained word embeddings of $FastText$ [2]. The decision to use $FastText$ is because it provides embeddings also for those words outside the training vocabulary.

2.2 Translation

In this stage, the pre-processed Q_{nat} is analyzed to recognize both its structure and the named entities that serve to build $S'_{Q_{nat}}$ in the subsequent assembling phase. To do such recognition, we employ two neural networks performing two key NLP tasks: (i) a Neural Machine Translation task (NMT), and (ii) a Named Entity Recognition (NER) task.

Table 1. $\langle Q_{nat}, Q_{sparql}\rangle$ for *Who painted the Mona Lisa?*

Question	**Query**
Who painted the Mona Lisa?	select ?a where { wd:Q12418 wdt:P170 ?a }

In the NMT task, Q_{nat} is translated in a *query template*. A query template is the skeleton of a SPARQL query where some of the KB resources are replaced by placeholders. In general, in an NMT task, given a source $X = (x_1, x_2, ..., x_n)$ sequence and a target $Y = (y_1, y_2, ..., y_m)$ sequence, the aim is to model the conditional probability of target words given the source sequence [1]. In our approach, an NMT neural network takes as input the preprocessed question and translates it into a SPARQL query template.

In the NER task, the entities present in Q_{nat} are identified and classified using a dedicated neural network. In a NER task (also known as *Entity Extraction*), named entities present in a text are associated with predefined categories, such as *individuals, companies, places, etc.* This additional semantic knowledge helps to understand the role of words in a given text [5]. As in most of the literature, in our implementation, we also adopt the *BIO* notation [9] to tag a text, which differentiates the beginning "*B*" and the interior "*I*" of the entities while "*O*" is used for non-entity tokens. The named entities identified by NER are combined with the output of NMT in the assembling phase to build $S'_{Q_{nat}}$.

Training set format. The NMT and NER networks are trained together using the same input, obtained by converting the original training set into a novel format called *QQT* format. This pre-processing step has a two-fold objective. On the one hand, it aligns the inputs of both NMT and NER tasks; and, on the other hand, it reduces the size of the output vocabulary and helps to mitigate the impact of the WOOV during the translation. Translating entities to URIs can be hard to learn from mere examples. A system can fail if the NER task is not simple and there are a lot of words that are out of the vocabulary of the training set.

A dataset in QQT is composed of a set of triples in the form $\langle$*Question*, $QueryTemplate, Tagging\rangle$, where *Question* is a natural language question, *Tagging* marks which parts of *Question* are entities, and *QueryTemplate* is a SPARQL query template modified as follows: (*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, while Table 2 shows the corresponding $\langle$*Question*, $QueryTemplate, Tagging\rangle$ triple in the QQT format. In Table 2, the term $1 denotes a placeholder where the number index 1 means that $1 has to be replaced by the first entity occurring in the question, *Mona Lisa* in this case, as represented by BIO tagging notation. Note that in the QQT format, the query template does not contain any KB resource, so the learning model does not need to understand *Mona Lisa* stands for a specific KB

Table 2. QQT triple for *Who painted the Mona Lisa?*

Question	**QueryTemplate**	**Tagging**
Who painted the Mona Lisa?	select ?a where { ?w wdt:P170 ?a . ?w rdfs:label $1 }	O O O B I O

resource identifier. With this representation, the output vocabulary of the NMT model is reduced, and the network is more tolerant to the WOOV problem.

The networks. As we have seen, the architecture has two important parts that are NMT and NER. To develop the NMT neural network, we decided to use the standard *Encoder-Decoder* approach, with BiLSTM and Luong Attention [8], which has shown high performance in the literature. On the other hand, to develop the NER network, we used the BiLSTM-CRF approach proposed by [7], which assigns a tag to each token in the input sequence. The two models share the fact that they have a BiLSTM-based encoder to obtain the semantic information from the input sequence. For this reason, we decided to do a joint training of the two models looking to improve the training times and make the two networks help each other.

The proposed approach, depicted in Figure 1 (*right*), has a single encoder composed of one BiLSTM layer with two branches connected to it. The first branch uses a CRF layer responsible for determining $p(l|x)$, which refers to the probability of calculating a tagging sequence l given an input sequence x. The second one is an NMT decoder, composed of one LSTM layer and the attention mechanism followed by a fully connected network calculating the probability $p(y|x)$ that the sequence y correctly translates x in $QueryTemplate$.

2.3 Assembling

The last step of the proposed architecture is the creation of Q'_{sparql}. Here the placeholders in Q'_{temp} are replaced by the corresponding named entities. For example, suppose we have the question-query pair:

```
<Who developed Skype?, SELECT ?uri WHERE { wd:Q40984 wdt:P178 ?uri } >
```

in the form $\langle Q_{nat}, Q_{sparql}\rangle$. Our system will translate Q_{nat} to the following query template:

```
SELECT ?uri WHERE { ?v wdt:P178 ?uri . ?v rdfs:label $1 }
```

and the NER task will identify that *Skype* is the named entity. So the query instantiation step will produce:

```
SELECT ?uri WHERE { ?v wdt:P178 ?uri . ?v rdfs:label "Skype"@en }
```

which is a SPARQL query equivalent to Q_{sparql}.

At this step, we validate the entities identified by the NER model to produce a more accurate query. The system checks that the *rdfs:label*-based triples created with the entities produce a non-empty result and consider them valid. If a triple does not have a match in the KB, the system attempts to fix it using the entities identified with the NEL tool. In this case, we pick the entity identified by NEL whose surface in the question overlaps with the entity we are trying to fix. If it is not possible to fix a non-valid triple, the system will probably respond with an incorrect answer.

3 Results

We have implemented our models using Keras, a well-known framework for machine learning, on top of TensorFlow. The training process was carried out on Google Colaboratory, a Jupyter notebook environment running entirely in the cloud. Colaboratory provides an environment with 12 GB of RAM and the possibility to run the code using a GPU configuration.

To train and fine-tune the system, we used the dataset released for the tenth edition of QALD[2]. Table 3 shows the results obtained by *sparql-qa* in the challenge, demonstrating that the system performed very well. The reported scores were calculated using the GERBIL-QA[3] web service. A live demo of the system is available at: https://www.mat.unical.it/ricca/sparqlqa.

Table 3. QALD-10 final scores.

	Macro Precision	**Macro Recall**	**Macro F1**	**Macro F1 QALD**
sparql-qa	0.4538	0.4574	0.4538	0.5947

Conclusion

The paper presents *sparql-qa*, an approach for querying knowledge bases by using natural language. The system relies on the combination of Neural Machine Translation and Named Entity Recognition while focusing on attenuating the impact of the OOV words. *sparql-qa* showed to be a valid proposition in the field of KBQA, and the results obtained in QALD-10 demonstrate this.

In future work, we plan to improve our system by integrating other NLP tools, such as BERT word embeddings, Transformers, and Relation Linking.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. TACL **5**, 135–146 (2017)
3. Borroto, M.A., Ricca, F., Cuteri, B.: Reducing the impact of out of vocabulary words in the translation of natural language questions into sparql queries. arXiv preprint arXiv:2111.03000 (2021)
4. Daiber, J., Jakob, M., Hokamp, C., Mendes, P.N.: Improving efficiency and accuracy in multilingual entity extraction. In: ICSS (I-Semantics) (2013)
5. Grishman, R., Sundheim, B.M.: Message understanding conference-6: A brief history. In: COLING 1996 Volume 1: The 16th Int. Conf. on Comp. Linguistics (1996)

[2] https://github.com/KGQA/QALD_10

[3] https://gerbil-qa.aksw.org/gerbil/experiment?id=202205200035

6. Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A.: spaCy: Industrial-strength Natural Language Processing in Python (2020)
7. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. CoRR **abs/1508.01991** (2015)
8. Luong, M., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
9. Ramshaw, L.A., Marcus, M.P.: Text chunking using transformation-based learning. In: Nat. lang. proc. using very large corpora, pp. 157–176. Springer (1999)
10. Shen, W., Wang, J., Han, J.: Entity linking with a knowledge base: Issues, techniques, and solutions. IEEE TKDE **27**(2), 443–460 (2014)
11. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS. pp. 3104–3112 (2014)