SPARQL as a Foreign Language

Tommaso Soru tsoru@informatik.uni-leipzig.de
Agile Knowledge Engineering and Semantic Web, Institute of Computer Science, University of Leipzig, Germany

Edgard Marx edgard.marx@htwk-leipzig.de
Faculty of Computer Science Mathematics and Natural Sciences, Leipzig University of Applied Sciences, Germany

Diego Moussallem moussallem@informatik.uni-leipzig.de
Agile Knowledge Engineering and Semantic Web, Institute of Computer Science, University of Leipzig, Germany

Gustavo Publio gustavo.publio@informatik.uni-leipzig.de
Agile Knowledge Engineering and Semantic Web, Institute of Computer Science, University of Leipzig, Germany

André Valdestilhas valdestilhas@informatik.uni-leipzig.de
Agile Knowledge Engineering and Semantic Web, Institute of Computer Science, University of Leipzig, Germany

Diego Esteves esteves@cs.uni-bonn.de
Smart Data Analytics, University of Bonn, Germany

Ciro Baron Neto cbaron@informatik.uni-leipzig.de
Agile Knowledge Engineering and Semantic Web, Institute of Computer Science, University of Leipzig, Germany

Keywords

Question Answering | Machine Translation | Neural Networks | Deep Learning | SPARQL | Linked Data

Abstract

Recently, the Linked Data Cloud has achieved a size of more than 100 billion facts pertaining to a multitude of domains. However, accessing this information has been significantly challenging for lay users. Approaches to problems such as Question Answering on Linked Data and Link Discovery have notably played a role in increasing information access. These approaches are often based on handcrafted and/or statistical models derived from data observation. Recently, Deep Learning architectures based on Neural Networks called seq2seq have shown to achieve the state-of-the-art results at translating sequences into sequences. In this direction, we propose Neural SPARQL Machines, end-to-end deep architectures to translate any natural language expression into sentences encoding SPARQL queries. Our preliminary results, restricted on selected DBpedia classes, show that Neural SPARQL Machines are a promising approach for Question Answering on Linked Data, as they can deal with known problems such as vocabulary mismatch and perform graph pattern composition.

Notes: Accepted at SEMANTiCS 2017 Posters & Demos. The first two authors contributed equally.

This version: http://w3id.org/neural-sparql-machines/soru-marx-semantics2017.html

Last version: http://w3id.org/neural-sparql-machines/soru-marx-semantics2017.html

1. Introduction

Nowadays, the Web has been growing notably and producing an enormous amount of information every day. Meanwhile, the Linked Data Cloud has been structuring these billions of facts from the Web. However, this enormous volume of data complicates the access of a given desired information. Thus, becoming a hard task for lay users that are not familiar with formal query languages such as SPARQL [1] to search their interests. To this end, Question Answering (QA) systems provide users with friendly interfaces that hide the complexity of creating such queries. These interfaces enable the users to perform questions using their natural language. Although the QA systems have shown great solutions for dealing with this huge amount of statements, QA still lacks good solutions for handling vocabulary mismatch and problems on graph patterns [2] [3] . Vocabulary mismatch occurs when the user types different terms of the ones contained in a Knowledge Base (KB) for querying a given information. Several works tried to alleviate this problem, such as [10] [2] [7] .

Recent advances in Deep Learning applied to Natural Language Processing tasks have shown a promising possibility of parsing, understanding, and translating language sentences. In addition, with the growth of embeddings-based techniques, the Machine Translation (MT) community brought back the application of Neural Networks (NN) within MT systems. The first Neural Machine Translation (NMT) approach was proposed by Kalchbrenner and Blunsom [5] . This approach implemented Deep Neural Networks to build end-to-end encoder–decoder models using embeddings, i.e. vector spaces, within the language models. NMT recently overcame the Phrased-based Machine Translation approach on the BLEU [9] score, thus rising the interest on NN applied to translation problems [13] . In NMT, pairs of sequences are given as input to a NN model, which is left to learn the translation model. Based on this formal description, we have an insight to translate from Natural Language (NL) to SPARQL by using NMT models in order to address the aforementioned QA gaps. Therefore, we propose Neural SPARQL Machine, an end-to-end learning model to translate any NL expression into a sequence of tokens which can reconstruct a SPARQL grammar. Our preliminary results show that Neural SPARQL Machines can perform compositions of new graph patterns which never occurred in the training set. In addition, when evaluated on BLEU, our Neural SPARQL Machine achieved an accuracy of 0.8.

2. Related work

QA approaches commonly convert NL questions into formal SPARQL queries in order to retrieve the answer from a *triple store* [12] [11] [2] . Approaches are designed to answer only facts, fact-based QA [7] , multiple facts Basic Graph Patterns [11] [15] or more complex queries [12] [2] . Most commonly, QA systems are based on a conjunction of previously-made rules derived from data analyses and observation [11] [15] [2] or learned based on positive and negative samples [12] [7] [14] . Therefore, the strategy used to generate the SPARQL graph patterns and consequently the SPARQL query varies significantly.

TBSL [12] is a template-based QA system that relies on SPARQL query templates to generate queries with structures that resemble the semantic structure of the original query. The template is detected based on learned positive and negative examples. The predicate and entity detection are performed by applying a *TF-IDF* function implemented in the *Lucene Apache Framework* [3] .

SINA [11] is a QA system that explores the knowledge base to formulate the SPARQL query by applying Jaccard similarity to evaluates possible resource matching candidates and Hidden Markov Models to choose the right answer among them. In this conversion, queries are identified (query-type) based on pre-defined syntactic rules and resources are linked using DBpedia Spotlight [4] . Recently, [7] proposed a fact retrieval approach based on a Recurrent Neural Network (RNN) encoding subject labels at character level and predicates and datatype values at both character and word levels.

3. Neural SPARQL Machine

We then propose a new paradigm for answering NL questions. Instead of using statistical and handcraft models, we shift the complete task of SPARQL query generation to Neural Simbolic Machines (NSM) which are a specific class of *seq2seq* models. NSMs aim at translating NL to a sequence of tokens defining a program by using a key-variable memory and reinforcement learning with weak supervision [6] . However, our approach for QA differs slightly from the one proposed by [6] , as the authors 1) train a NN on query templates

to avoid over-fitting and 2) use reinforcement learning to optimize the task reward of the problem. We instead use an end-to-end approach that translates an entire NL expression into a final query (see an example in Figure 1).

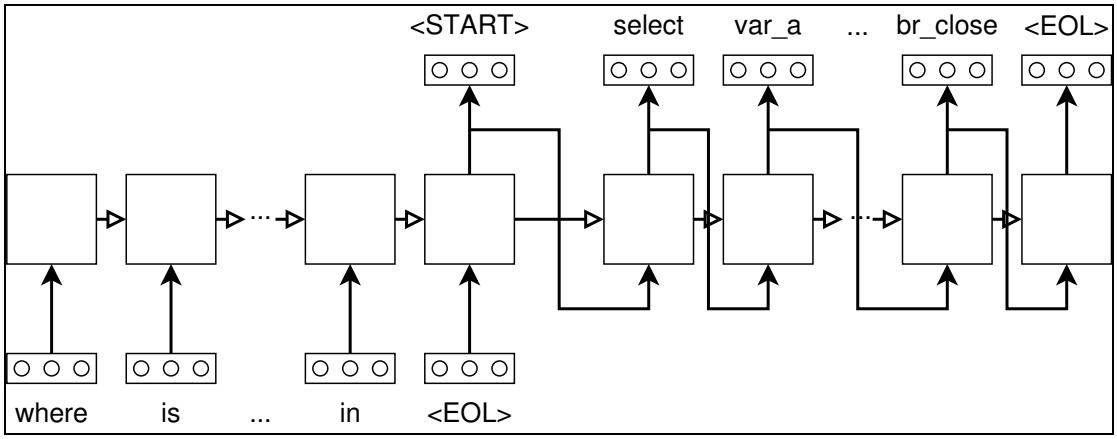

Figure 1. LSTM architecture for sequence-to-sequence learning of a machine translator from natural language to SPARQL. The left side is the *encoder* , whereas the right side shows the *decoder* . The question *"Where is Hill 60 located in?"* is translated into the sequence of tokens: `select var_a where br_open dbr_Hill_60_(Ypres) dbo_location var_a br_close` .

3.1. Architecture

The architecture of our Neural SPARQL Machine relies on three main components: a *generator* , a *learner* , and an *interpreter* . The generator takes query templates as input and creates the training dataset which will be forwarded to the learner. A *query template* is an alignment between a natural language query and its respective SPARQL query, with entities replaced by placeholders. The learner takes natural language as input and generates a sequence which encodes a SPARQL query. The final structure is then reconstructed by the interpreter.

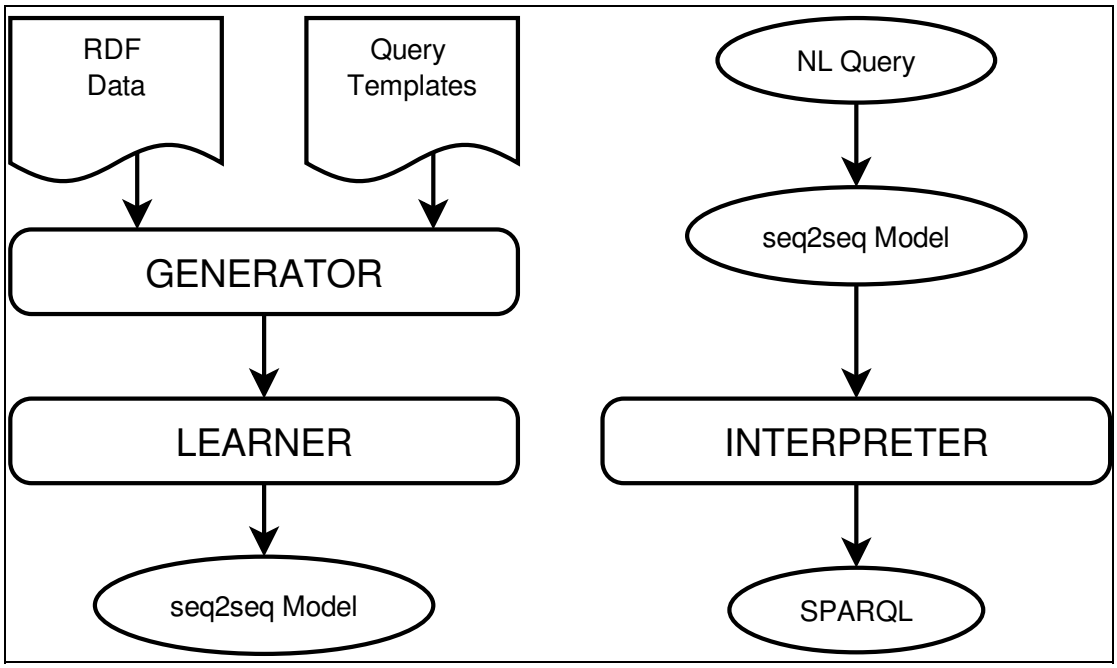

Figure 2. Architecture of a Neural SPARQL Machine at training phase (left) and prediction phase (right).

An overview of the architecture can be seen in Figure 2 , where we highlighted the differences between the training and the prediction phases of the machine-learning workflow. Below are a few examples of query

templates:

```
where is <A> located in?
  ->  select ?a where { <A> dbo:location ?a }
what are the <A> northernmost <B>?
  ->  select ?a where { ?a rdf:type <B> . ?a geo:lat ?b }
      order by desc(?b) limit <A>
was <A> finished before <B>?
  ->  ask where { <A> dbp:complete ?a . FILTER(?a <= <B>) }
```

3.2. SPARQL Modeling

We focus on learning a model which is suited for a specific KB and therefore we do not expect to reuse a learned model for QA on other KBs. The rationale behind this viewpoint complies the concept of the Semantic Web, which allows users to define their own classes, instances, and properties without a need of a global design. For instance, a model trained on DBpedia may learn that *"is located in"* translates to `dbo:country`. However, property `dbo:country` might not have any sense in another dataset, where another property is used instead. For example, the property `http://schema.org/Country` used in the Schema.org ontology.

Each of the SPARQL operators (e.g., `SELECT`, `FILTER`, or `ORDER BY`) in a query is encoded using a certain number of tokens. We decided to encode brackets, wildcards, and dots, since they are also used inside the query. URIs are shortened using prefixes and column characters replaced with underscores to keep each URI as a single token.

3.3. The Learner

Our learner features a *seq2seq* model with Long Short-Term Memory (LSTM). The seq2seq model takes a sequence of word embeddings X = { x 1 , x 2 , ... , x M } as input and outputs a translated sequence Y = { y 1 , y 2 , ... , y N } . Aim of the model is to maximize the generation probability of *Y* conditioned on *X* , i.e. p (y 1 , ... , y N | x 1 , ... , x M) . As seen in Figure 1 , the model is composed by a encoder and a decoder. Using a RNN, an encoder *q* reads the input words into a vector:

$$c = q (h 1 , ... , h T) . (1)$$

The hidden states at a time *t* of the NN are computed as follows:

$$h t = f (x t , h t - 1) (2)$$

where f is a non-linear transformation performed by a LSTM using sigmoid function. The same workflow applies specularly for the decoder, after which the output sequence *Y* is outputted.

The plain version of our learner learns word embeddings from the input corpus. However, as previously proposed in [6] , word embeddings can also be imported from a pre-trained corpus of Word2Vec [8] vectors to help tackling known QA problems such as vocabulary mismatch.

4. Preliminary results

4.1. Data generation

We use DBpedia as a use-case for our preliminary test, considering the extension of one class as working domain, i.e. `dbo:Monument`. The class extension contains 35,095 triples and 625 instances. To generate the training data, we manually annotated 38 query templates involving one or two entities per query. For each of these templates, we fetched from DBpedia a given number of examples that satisfied the corresponding SPARQL graph patterns. For instance, for question *"where is <A> located in?"* , we fetched only instances

having a dbo:location property. These instances were randomly extracted and added to the working dataset. We split the working dataset into three parts: training, validation, and test dataset. We fixed the size of the validation and test sets to 100 examples.

4.2. Experiments

We carried out an evaluation of the BLEU accuracy [9] for the NMT model. BLEU uses a modified precision metric for comparing the MT output with the reference translation, in our case, SPARQL queries. As Table 1 shows, the first setting expects to have an average of 13.35 examples per instance, meaning that the model has — on average — around 13-14 times to learn that certain labels translate into a certain URI.

Dataset	Examples per template	Training size	Average examples per instance
dbo:Monument	300	8,344	13.35
dbo:Monument	600	14,588	23.34

Table 1. Information about the experiments carried out on the datasets.

The experiments were carried out on a 64-core CPU-only Ubuntu machine with 512 GB RAM. [4] We adopted the implementation of seq2seq in TensorFlow [1] with embeddings of 128 dimensions, 2 hidden layers, and a dropout value of 0.2. We tested our NMT model at three different times, i.e. at the 12,000th, 36,000th, and 120,000th iteration.

Dataset	Examples per template	BLEU 12k epochs	BLEU 36k epochs	BLEU 120k epochs
dbo:Monument	300	0.753 (0:18:24)	0.767 (0:55:28)	0.765 (3:05:43)
dbo:Monument	600	0.801 (0:19:10)	0.803 (0:58:09)	0.803 (3:11:44)

Table 2. BLEU accuracy after a given training time (expressed as H:MM:SS) for different number of examples per query template.

The complete evaluation results are shown in Table 2 . One of the main drawbacks of deep-learning architectures is the computational complexity and training time. As can be seen, the number of examples per template does not seem to affect the training time as much as the number of iterations. However, the BLEU accuracy achieved with *n=600* saw an increase of *+5%* , which can lead to think that more examples might further improve the translation. As Figure 3 shows, this value stabilizes after 10,000 epochs, which correspond to ~16 minutes of runtime. Minimizing the number of examples per instance, we estimate to learn a model on a large dataset as DBpedia in less than 20 days; GPU architectures can bring this value down to 5 days. Since the BLEU accuracy gives only an estimation of how the alignments are performed between the languages, we plan to use standard measures such as F-score to evaluate the final prediction of SPARQL queries.

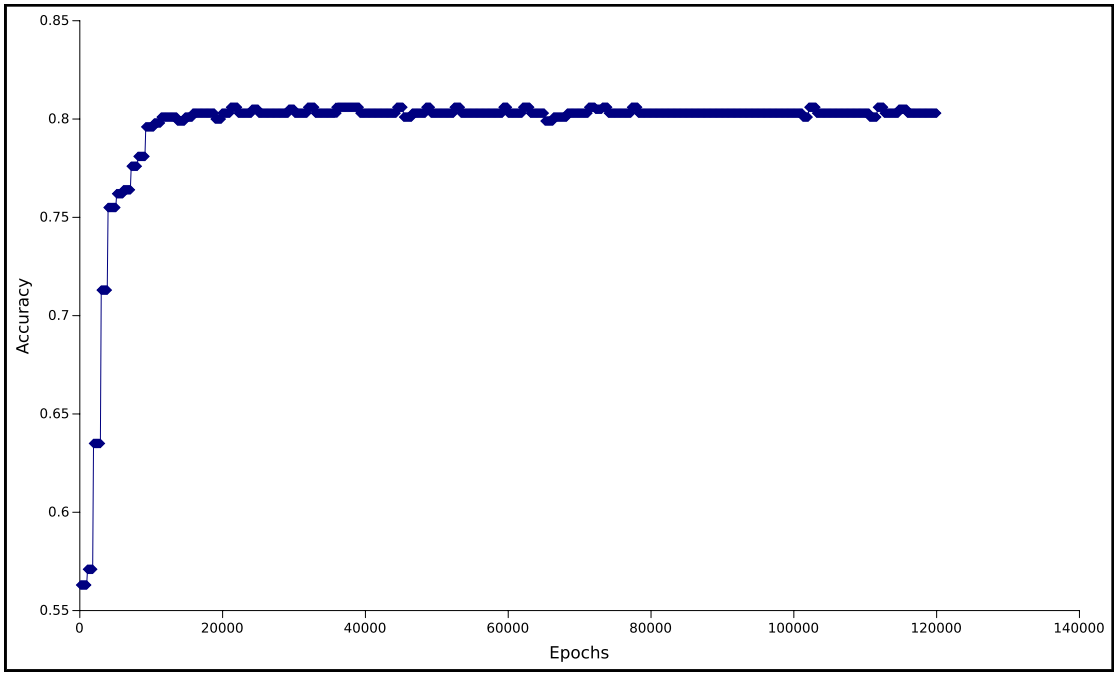

Figure 3. BLEU accuracy on the test set for the translations on dataset `dbo:Monument` with 600 examples per query template.

To test compositionality, we inputted complex questions to our Neural SPARQL Machine. An example is *"where are the 3 northernmost monuments located in?"* . Although the model never saw such a question before, it was able to output a sequence which includes the features of two learned templates: `select var_a where br_open var_a rdf_type dbo_Monument sep_dot var_a geo_lat var_b sep_dot var_a dbo_location var_b br_close order_by desc var_b limit 3` . Still, the interpreter will need to replace variables `var_a` or `var_b` with a new variable `var_c` to fetch the desired information. This is an open challenge for our model.

5. Conclusion and future work

In this work, we presented a neural approach for the translation of NL questions in SPARQL queries using a NN model. Preliminary results show an average BLEU accuracy of ~0.8 when trained on QA template model pairs, even when answering such questions requires the formulation of more complex queries. The current model is vocabulary-dependent and needs to be trained on samples derived from manually-created QA template pairs. We plan to address current limitations by investigating how to generate domain-independent templates and minimize the burden on the end user.

References

1. Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., Corrado G., Davis A., Dean J., Devin M., others (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.
2. Dubey M., Dasgupta S., Sharma A., Hoffner K., Lehmann J. (2016). AskNow: A Framework for Natural Language Query Formalization in SPARQL. The Semantic Web. Latest Advances and New Domains: 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 -- June 2, 2016, Proceedings.
3. Hoffner K., Walter S., Marx E., Usbeck R., Lehmann J., Ngonga Ngomo A. (2016). Survey on challenges of Question Answering in the Semantic Web. Semantic Web.
4. Joachim Daiber, Max Jakob, Chris Hokamp, Pablo N. Mendes (2013). Improving Efficiency and Accuracy in Multilingual Entity Extraction. UNKNOWN.

5. Kalchbrenner N., Blunsom P. (2013). Recurrent Continuous Translation Models. EMNLP.
6. Liang C., Berant J., Le Q., Forbus K., Lao N. (2016). Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. arXiv preprint arXiv:1611.00020.
7. Lukovnikov D., Fischer A., Auer S., Lehmann J. (2017). Neural Network-based Question Answering over Knowledge Graphs on Word and Character Level. Proceedings of the 26th international conference on World Wide Web.
8. Mikolov T., Sutskever I., Chen K., Corrado G., Dean J. (2013). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems.
9. Papineni K., Roukos S., Ward T., Zhu W. (2002). BLEU: a method for automatic evaluation of machine translation. Proceedings of the 40th annual meeting on association for computational linguistics.
10. Shekarpour S., Marx E., Auer S., Sheth A. (2017). RQUERY: Rewriting Natural Language Queries on Knowledge Graphs to Alleviate the Vocabulary Mismatch Problem. AAAI.
11. Shekarpour S., Marx E., Ngomo A., Auer S. (2015). {SINA}: Semantic Interpretation of User Queries for {Q}uestion {Answering} On Interlinked Data. Journal of Web Semantics.
12. Unger C., Buhmann L., Lehmann J., Ngonga Ngomo A., Gerber D., Cimiano P. (2012). Template-based Question Answering over RDF Data. Proceedings of the 21st International Conference on World Wide Web.
13. Wu Y., Schuster M., Chen Z., Le Q., Norouzi M., Macherey W., Krikun M., Cao Y., Gao Q., Macherey K., others (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.
14. Yih W., Chang M., He X., Gao J. (2015). Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015).
15. Zhang Y., He S., Liu K., Zhao J. (2016). A Joint Model for Question Answering over Multiple Knowledge Bases. Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence.

Footnotes

1 http://www.w3.org/TR/rdf-sparql-query/ [back]

2 For a definition of graph patterns, please visit https://www.w3.org/TR/rdf-sparql-query/ . [back]

3 https://lucene.apache.org [back]

4 Source code and datasets available at http://w3id.org/neural-sparql-machines/code/ . [back]