Neural Machine Translation for Query Construction and Composition

Tommaso Soru 1  Edgard Marx 1  André Valdestilhas 1  Diego Esteves 2  Diego Moussallem 1  Gustavo Publio 1

Abstract

Research on question answering with knowledge base has recently seen an increasing use of deep architectures. In this extended abstract, we study the application of the neural machine translation paradigm for question parsing. We employ a sequence-to-sequence model to learn graph patterns in the SPARQL graph query language and their compositions. Instead of inducing the programs through question-answer pairs, we expect a semi-supervised approach, where alignments between questions and queries are built through templates. We argue that the coverage of language utterances can be expanded using late notable works in natural language generation.

1. Introduction

Question Answering with Knowledge Base (KBQA) parses a natural-language question and returns an appropriate answer that can be found in a knowledge base. Today, one of the most exciting scenarios for question answering is the Web of Data, a fast-growing distributed graph of interlinked knowledge bases which comprises more than 100 billions of edges (McCrae et al., 2018). Question Answering over Linked Data (QALD) is a subfield of KBQA aimed at transforming utterances into SPARQL queries (Lopez et al., 2013). Being a W3C standard, SPARQL features a high expressivity (Prud’hommeaux et al., 2006) and is by far the most used query language for Linked Data.

Among traditional approaches to KBQA, Bao et al. (2014) proposed question decomposition and Statistical Machine Translation to translate sub-questions into triple patterns. The method however relies on entity detection and struggles in recognizing predicates by their contexts (e.g., play in a film or a football team). In the last years, several methods based on neural networks have been devised to tackle the KBQA problem (Liang et al., 2016; Hao et al., 2017; Lukovnikov et al., 2017; Sorokin & Gurevych, 2017). We study the application of the Neural Machine Translation paradigm for question parsing using a sequence-to-sequence model within an architecture dubbed Neural SPARQL Machine, previously introduced in Soru et al. (2017). Similarly to Liang et al. (2016), we employ a sequence-to-sequence model to learn query expressions and their compositions. Instead of inducing the programs through question-answer pairs, we expect a semi-supervised approach, where alignments between questions and queries are built through templates. Although query induction can save a considerable amount of supervision effort (Liang et al., 2016; Zhong et al., 2017), a pseudo-gold program is not guaranteed to be correct when the same answer can be found with more than one query (e.g., as the capital is often the largest city of a country, predicates might be confused). On the contrary, our proposed solution relies on manual annotation and a weakly-supervised expansion of question-query templates.

2. Neural SPARQL Machines

Inspired by the Neural Programmer-Interpreter pattern by (Reed & De Freitas, 2015), a Neural SPARQL Machine is composed by three modules: a generator, a learner, and an interpreter (Soru et al., 2017). We define a query template as an alignment between a natural language question and its respective SPARQL query, with entities replaced by placeholders (e.g., “where is <A> located in?”). The gen-
Table 1. Experiments on a DBpedia subset about movies with different SPARQL encodings and settings.

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Description</th>
<th>Test BLEU</th>
<th>Accuracy</th>
<th>Runtime (h:m:s)</th>
<th>Convergence</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>1:1 SPARQL encoding</td>
<td>80.89%</td>
<td>22.33%</td>
<td>1h02:01</td>
<td>13,000</td>
</tr>
<tr>
<td>v1.1</td>
<td>Improved consistency</td>
<td>80.61%</td>
<td>22.33%</td>
<td>1h21:21</td>
<td>17,000</td>
</tr>
<tr>
<td>v2</td>
<td>Added templates with &gt; 1 placeholders</td>
<td>89.69%</td>
<td>91.04%</td>
<td>1h59:10</td>
<td>22,000</td>
</tr>
<tr>
<td>v2.1</td>
<td>Encoding fix (double spaces removed)</td>
<td>98.40%</td>
<td>91.05%</td>
<td>1h47:11</td>
<td>20,000</td>
</tr>
<tr>
<td>v3</td>
<td>Shortened SPARQL sequences</td>
<td>99.28%</td>
<td>94.82%</td>
<td>1h12:07</td>
<td>25,000</td>
</tr>
<tr>
<td>v4</td>
<td>Added direct entity translations</td>
<td>99.29%</td>
<td>93.69%</td>
<td>1h23:00</td>
<td>20,000</td>
</tr>
</tbody>
</table>

Neural SPARQL Machines do not rely on entity linking methods, since entities and relations are detected within the query construction phase. External pre-trained word embeddings help deal with vocabulary mismatch. Knowledge graph jointly embedded with SPARQL operators (Wang et al., 2014) can be utilized in the target space. A curriculum learning paradigm can learn graph pattern and SPARQL operator composition, in a similar fashion of Liang et al. (2016). We argue that the coverage of language utterances can be expanded using techniques such as Question (Abujabal et al., 2017; Elsahar et al., 2018; Abujabal et al., 2018) and Query Generation (Zafar et al., 2018) as well as Universal Sentence Encoder (Cer et al., 2018). Another problem is the disambiguation between entities having the same surface forms. Building on top of the DBtrends approach (Marx et al., 2016), we force the number of occurrences of a given entity in the training set to be inversely proportional to the entity ranking. Following this strategy, we expect the RNN to associate the word Berlin with the German capital and not with Berlin, New Hampshire.

3. Experiments and current progress

We selected the DBpedia Knowledge Base (Lehmann et al., 2015) as the dataset for our experiments, due to its central importance for the Web of Data. We built a dataset of 3,108 entities about movies from DBpedia and annotated 20 and 4 question-query templates with one and two placeholders, resp. Our preliminary results are given in Table 1.

We experimented with 6 different SPARQL encodings, i.e. ways to encode a SPARQL query into a sequence of tokens. At each row of the table, we provide the description of the corresponding changes, each of which persists in the next encodings. The experiments were carried out on a 64-CPU Ubuntu machine with 512 GB RAM. We adopted the implementation of seq2seq in TensorFlow with internal embeddings of 128 dimensions, 2 hidden layers, and a dropout value of 0.2. All settings were tested on the same set of unseen questions after applying an 80-10-10% split.

The results confirmed that the SPARQL encoding highly influences the learning. Adding more complex templates (i.e., with more than one placeholder) to the generator input yielded a richer training set and more questions were parsed correctly. Merging tokens (see queries and their respective sequences in Figure 1) helped the machine translation, as the SPARQL sequences became shorter. Adding alignments of entities and their labels to the training set turned out to be beneficial for a faster convergence, as Figure 2 shows. The most frequent errors were due to entity name collisions and out-of-vocabulary words; both issues can be tackled with the strategies introduced in this work.

We plan to perform an evaluation on the WEBQUESTION-SSP (Yih et al., 2016) and QALD (Unger et al., 2014) benchmarks to compare with the state-of-the-art approaches for KBQA and QALD, respectively.

Figure 2. BLEU accuracy against training epochs.

Code available at https://github.com/AKSW/NSpM.
References


