Mitigating the impact of out of vocabulary words in a neural-machine-translation-based question answering system

Manuel Borroto[1], Bernardo Cuteri[1][0000−0001−5164−9123], and Francesco Ricca[1][0000−0001−8218−3178]

University of Calabria, Rende CS 87036, Italy
{manuel.borroto,bernardo.cuteri,francesco.ricca}@unical.it
https://informatica.unical.it

Abstract. The diffusion of ontologies ended up in the development of rich knowledge bases featuring large volumes of information concerning multiple domains. However, the largest majority of potential users are unfamiliar with the SPARQL query language, and thus can enjoy only limited access to knowledge bases provided by predefined interfaces. Systems able to translate questions posed in natural language in SPARQL queries have the potential of overcoming this problem. In this paper, we approach this problem as a Neural Machine Translation task to implement an automatic translation of natural language questions in SPARQL queries. A distinctive feature of our deep-learning-based approach is its robustness with respect to the presence of terms (referring to individuals) that do not occur in the training set. We demonstrate the potential of our approach by presenting its results on the Monument dataset, a benchmark for Question Answering on the well-known DBpedia ontology.

Keywords: Natural Language Processing · Question Answering · Knowledge base · Neural Machine Translation.

1 Introduction

The diffusion of ontologies as a mean for modeling, storing, and sharing information determined the development of rich knowledge bases featuring large volumes of information concerning multiple domains. As a result of this, we now have vast and complex knowledge bases that allow gathering large volumes of information through the intercommunication of thousands of datasets referring to various domains in what is known as Linked Data. Thus, people have potential access to a large amount of information never thought, and the DBpedia [11] project is a real example of that, which is one of the most popular knowledge bases nowadays. However, 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 [19]. This means that the largest majority of users have only limited access to knowledge bases as it is provided by predefined interfaces.

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. Thus, natural language Question Answering (QA) is gaining importance in the area of the Semantic Web.

The most recent QA approaches resulted [3, 14, 16] in systems for the automatic translation from natural language questions to SPARQL queries. These are mostly based on deep neural networks to tackle the problem and exploit the great development achieved by Deep Learning in the last few years.

In this paper, we approach this problem as a Neural Machine Translation task to implement an automatic translation of natural language questions in SPARQL queries. Our system was built to be robust with respect to the presence of terms (referring to individuals) that do not occur in the training set, also called *out of vocabulary words*. A feature particularly useful when dealing with evolving ontologies that are continuously enriched with new individuals.

We achieve this result with a novel architecture that combines based on a Neural Machine Translation (NMT) [1] module and a Named Entity Recognition (NER) module both based on bidirectional recurrent neural networks [18, 10]. 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 results of the two modules results in a SPARQL query ready to be executed. Importantly, we introduce a formal definition of a training set format that reduces the output space and is essential for the proper functioning of the system and also allows us to tackle the problem with out-of-vocabulary (OOV) words, a major weakness of the majority of the related approaches today. We empirically test the system on the Monument dataset[8], which is a benchmark for Question Answering on the well-known DBpedia ontology.

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

In the following, we assume the reader already knows the main concepts and techniques applied in the content of this research work, such as Knowledge bases, Neural Networks, Deep Learning, Natural Language Processing, Neural Machine Translation, Named Entity Recognition, among others. To go more in deep into these topics, please refer to [7, 5, 1, 9, 4].

2 From 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 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 $answers(S'_{Q_{nat}}) = answers(G_{Q_{nat}})$.[1] 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*.

Most of the solutions currently proposed to convert from natural language to SPARQL language make use of various techniques, either using patterns or deep neural networks. In any machine translation technique, the definition of input and output vocabularies is necessary, which, working with natural language, can become large enough to be a real problem when undertaking the translation task. This large size directly affects systems based on neural networks because they depend on a training set that allows networks to generalize a given domain. Obtaining a good dataset that includes all the words and names in the English language and includes all DBpedia resources is a task with a high level of difficulty. The datasets currently available comprise only a part of the vocabulary, generating a problem of *Words Out Of Vocabulary* (WOOV) that affects both the input and the output.

Systems affected by the WOOV problem have difficulty dealing with words not seen during the training phase because they do not know how to map those words to the output vocabulary. For example, let's assume we have a training set containing the *"Abraham Lincoln"* words and a system trained on it. If we want to translate the question *When Abraham Lincoln was born?*; the system will be able to identify the right KB resource, but on the other hand, the system will fail to translate a question using the same pattern, but changing *"Abraham Lincoln"* by something not present in the vocabulary, let say *"Barack Obama"*.

To reduce the impact of the WOOV and to boost the training time of the entire process, we will introduce in the next subsection a suitable format to represent an NL to SPARQL datasets that we call *QQT* format.

[1] Note that we are interested in computing the answers, and not in reproducing syntactically the gold query.

Table 1. $\langle Question, Query\rangle$ pair for *Who painted the Mona Lisa?*

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

2.1 A data set format for mitigating the WOOV problem

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 $\langle Question, QueryTemplate, Tagging\rangle$, 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 $\langle Question, QueryTemplate, Tagging\rangle$ 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 Mona Lisa stands for the *dbr:Mona_Lisa* resource and the $QueryTemplate$ is exactly the same for all questions asking the author of a given artwork.

2.2 Out-of-vocabulary words in NL questions

Although we can reduce the size of the output vocabulary by creating a QQT dataset, there is still a problem with the input vocabulary because there may be many absent words. This problem causes the model not to learn how to translate those OOV words because they were not seen during the training process.

Table 2. $\langle Question, QueryTemplate, Tagging\rangle$ pair 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

To address this problem, we used the pre-trained word embeddings provided by the FastText[2] library, allowing us to have access to thousands of embeddings vectors learned over millions of words, becoming a positive aspect because to obtain something similar, it is necessary a lot of time and computational resources. FastText can provide a word-embedding of a token even if it was not part of the vocabulary used to train the vectors, making it possible to manipulate OOV words.

2.3 The Model

Our approach consists of two deep neural networks, the first one specialized in Neural Machine Translation (NMT) based on the well-known Seq2Seq[18] model and the second one used for extracting the entities from the question using the Named Entity Recognition (NER) technique.

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* [12], 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, in the form $\{x_1, x_2, ..., x_t\}$, where x_t is the vector representation of the word t in the sentence. Next, we use a Bidirectional LSTM (BiLSTM) to summarize $\{x_1, x_2, ..., x_t\}$ 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.

Named Entity Recognition To perform the entity recognition, we created a BiLSTM-CRF [10] 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.

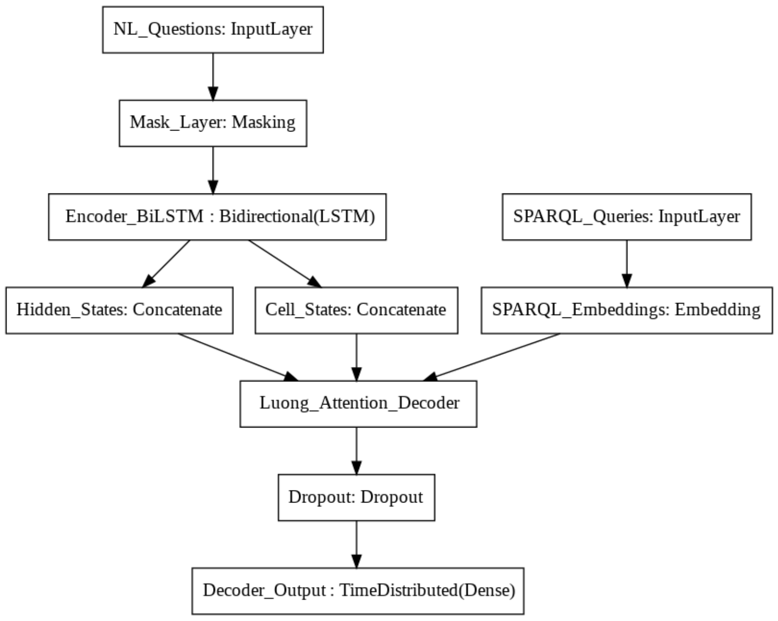
NL_Questions: InputLayer
Mask_Layer: Masking
Encoder_BiLSTM : Bidirectional(LSTM)
SPARQL_Queries: InputLayer
Hidden_States: Concatenate
Cell_States: Concatenate
SPARQL_Embeddings: Embedding
Luong_Attention_Decoder
Dropout: Dropout
Decoder_Output : TimeDistributed(Dense)

Fig. 1. NMT neural network architecture

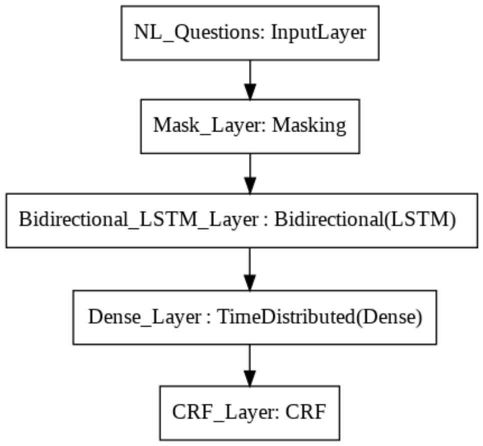
NL_Questions: InputLayer
Mask_Layer: Masking
Bidirectional_LSTM_Layer : Bidirectional(LSTM)
Dense_Layer : TimeDistributed(Dense)
CRF_Layer: CRF

Fig. 2. NER neural network architecture

3 Experiments

We report here an empirical assessment of our approach.

Experiment Setup. We have implemented our models by using Keras, a well-known framework for machine learning, on top of TensorFlow.We trained the

	Mon300			**Mon600**		
	P	**R**	**F1**	**P**	**R**	**F1**
NSpM	0.860	0.861	0.852	0.929	0.945	0.932
sparql-qa	0.78	0.78	0.78	0.791	0.791	0.791

Table 3. Comparison on Monument datasets.

networks by using Google Collaboratory, which is a virtual machine environment hosted in the cloud and based on Jupyter Notebooks. The environment provides 12GB of RAM and connects to Google Drive. We considered a well-known publicly available dataset for QA over the DBpedia ontology: the Monument dataset. To assess the systems, we adopted the macro precision, recall, and F1-score measures, which are the most used ones to assess this kind of system.

3.1 Evaluation on Monument dataset

The Monument dataset was proposed as part of the Neural SPARQL Machines (NSpM) [16] research. It contains 14,778 question-query pairs about the instances of type monument present in DBpedia.

For the sake of comparison with the state-of-the-art, we have trained the Learner Module of NSpM as it was done in [16], 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.[2] 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. From the results of the execution reported in Table 3, we can see that our system performs reasonably well, reaching F1-score values greater than 0.7. On the other hand, NSpM achieves better results.

We have investigated the cases in which our system could not provide an optimal answer, and we discovered that the performance of our approach is mainly affected by problems in the dataset. We found 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, NSpM can often use the specific URI of the gold query.

[2] https://github.com/LiberAI/NSpM/tree/master/data

Table 4. Comparison with OOV entities on Monument datasets.

	Mon300			**Mon600**		
	P	**R**	**F1**	**P**	**R**	**F1**
NSpM	0.097	0.123	0.101	0.11	0.11	0.11
sparql-qa	0.795	0.795	0.795	0.785	0.785	0.785

Thus, we decided to devise a tougher experiment to better understand the issue. We used the templates provided by NSpM and a randomly selected set of unseen monument entities extracted from DBpedia to create a new test set of 200 pairs. The results reported in Table 4 show that our approach confirms the same good performance (F1 score greater than 0.78), demonstrate to be capable of better generalizing power being basically resilient to the presence of unseen entities, and also performs better than NSpM.

4 Related Work

Pattern-based. The idea of employing query patterns for mapping questions to SPARQL-queries was already exploited in the literature [15, 17]. The approach presented by Pradel and Ollivier [15] 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 [17] 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 make 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 [23] 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 was proposed also in the literature [13] obtaining good results.

The *Neural SPARQL Machines (NSpM)* [16] 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*[8], and their model was tested on a subdomain referring to monuments and evaluated using the purely syntactic BLEU score [16]. 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. [3] 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. [3] 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 [20]. In this large experiment, the ConvS2S [6] 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 [21] 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 [22]. They implement a Seq2Seq model with Luong's attention, using BiLSTMs and BERT embeddings. The approach demonstrates to perform well on SParC and Spider datasets, outperforming the related work in some cases.

Our architecture addresses many of 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.

5 Conclusions and Future Work

The paper presents an approach based on deep neural networks to interrogate knowledge bases by using natural language. We exploit the strength of several well-known NLP tools and pose a special focus on reducing the target vocabulary of the NMT task and attenuating the impact of the OOV words, an important issue that is not well considered in existing approaches. Our system showed competitive results on the Monument dataset and demonstrated a more general and robust behavior on unseen questions among the compared system. 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 run our experiments to other well-known QA benchmarks.

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. Chen, Y., Li, H., Hua, Y., Qi, G.: Formal query building with query structure prediction for complex question answering over knowledge base. In: IJCAI (2020)
4. Cho, K., V. M., B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv:1406.1078 (2014)
5. Francois, C.: Deep learning with Python. Manning Publications Company (2017)
6. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: ICML. Proc. of ML Research, vol. 70, pp. 1243–1252. PMLR (2017)
7. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing? Int. J. Hum.-Comput. Stud. **43**(5-6), 907–928 (1995)
8. Hartmann, A., Marx, E., Soru, T.: Generating a large dataset for neural question answering over the DBpedia knowledge base (2018)
9. Hochreiter, S.: Recurrent neural net learning and vanishing gradient. Intern. Jour. Of Uncert., Fuzz. and KB Systems **6**(2), 107–116 (1998)
10. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. CoRR **abs/1508.01991** (2015)
11. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., et al.: Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web **6**(2), 167–195 (2015)
12. Luong, M., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
13. Luz, F.F., Finger, M.: Semantic parsing natural language into SPARQL: improving target language representation with neural attention. CoRR **abs/1803.04329** (2018)
14. Panchbhai, A., Soru, T., Marx, E.: Exploring sequence-to-sequence models for sparql pattern composition. In: Iberoamerican Knowledge Graphs and Semantic Web Conference. pp. 158–165. Springer (2020)
15. Pradel, C., Haemmerlé, O., Hernandez, N.: Natural language query interpretation into sparql using patterns (2013)
16. Soru, T., Marx, E., Moussallem, D., Publio, G., Valdestilhas, A., Esteves, D., Neto, C.B.: SPARQL as a foreign language. SEMANTiCS 2017 - Posters and Demos (2017), `https://arxiv.org/abs/1708.07624`
17. Steinmetz, N., Arning, A., Sattler, K.: From natural language questions to SPARQL queries: A pattern-based approach. In: BTW. LNI, vol. P-289, pp. 289–308. Gesellschaft für Informatik, Bonn (2019)
18. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS. pp. 3104–3112 (2014)
19. W3C: Semantic web standards (2014), `https://www.w3.org`
20. Yin, X., Gromann, D., Rudolph, S.: Neural machine translating from natural language to SPARQL. CoRR **abs/1906.09302** (2019)
21. Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., 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)
22. Zhang, R., Yu, T., Er, H.Y., Shim, S., Xue, E., Lin, X.V., Shi, T., Xiong, C., Socher, R., Radev, D.: Editing-based sql query generation for cross-domain context-dependent questions. arXiv preprint arXiv:1909.00786 (2019)
23. Zhong, V., Xiong, C., Socher, R.: Seq2sql: Generating structured queries from natural language using reinforcement learning. CoRR **abs/1709.00103** (2017)