A Sequence to Sequence approach for Knowledge Base Relation Linking

Vito Barbara[1], Manuel Borroto[1], and Francesco Ricca[1][0000−0001−8218−3178]

University of Calabria, Rende CS 87036, Italy
vitobarbara3@gmail.com, {manuel.borroto,francesco.ricca}@unical.it
https://informatica.unical.it

Abstract. Currently, *Question Answering* (QA) has taken a central role in the area of the *Semantic Web*, allowing access to a large amount of information in knowledge bases in a simple way. In this field, this is known as Knowledge Base Question Answering (KBQA). The most recent KBQA systems allow to abstract all those lay users from the complexities of the query languages like SPARQL and the ontology definition by posing questions in natural language to retrieve information. *Relation Linking* (RL) is an essential component in end-to-end KBQA systems, allowing to map relations in natural language questions to corresponding relations in knowledge bases. In this paper, we propose an approach for RL that leverages the strength of the *Deep Learning Sequence to Sequence models*. We demonstrate the potential of our system by presenting its results on *DBNQA* and QALD-9, which are well-known datasets for Question Answering over DBpedia.

Keywords: Relation Linking · Knowledge Base · Natural Language Processing.

1 Introduction

The development of knowledge bases has gathered nowadays large volumes of information concerning multiple domains. Among the best known is *DBpedia* [13], which covers various domains of interest. Unfortunately, access to this information is complicated for those users unfamiliar with the SPARQL [26] query language and the knowledge base definition, and so the *Knowledge Base Question Answering systems* have been introduced, as [4,3,12]. *KBQA* is a discipline that permits to answer automatically to natural language questions, retrieving information from knowledge bases. *RL* is a fundamental component in a *KBQA system*. Its goal is to associate each relation in a natural language question with the corresponding predicate in the knowledge base. For instance, given the natural language question *"How tall is the Eiffel Tower?"* an ideal RL system should return the predicate "dbo:height". This task is made particularly difficult due

to a large number of predicates in knowledge bases and the syntactic difference between the relations in natural language questions and the relations in knowledge bases. The possibility of multiple or implicit relations for a single question makes the work more complicated. Our idea is to treat the relations as a sequence of predicates, such that the RL task can be seen as a *sequence to sequence* [21] problem, where the input is a question, and the output is a sequence of relations. To this aim, we propose a system called *SeqRL*. The considered knowledge base is DBpedia, so any reference to knowledge bases is related to it. However, our approach is quite flexible: if it is necessary to use another knowledge base, it is enough only to change the output vocabulary and re-train the model on a new appropriate dataset.

We empirically test the system on two datasets for question answering on the DBpedia ontology, namely the DBpedia Neural Question Answering dataset (DBNQA) [9], and QALD-9 [16].

The remainder of the paper is structured as follows. In section 2, we talk about some related works. Section 3 provides the preliminary elements necessary to understand how our approach works. In section 4, we go into the particular details of our approach. Section 5 focuses on the discussion of experiments and results. And finally, in section 6, we provide some conclusions and aspects for future work.

2 Related work

Many question answering systems over knowledge graphs rely on relation linking components in order to do a proper mapping between the natural language question and the underlying knowledge graph. In this sense, Dubey et al.[5] proposed the *EARL* system, introducing the novelty that Entity Linking (EL) and Relation Linking tasks are done jointly, improving not only effectiveness but also execution times. EARL proposes two ways to face the problem. The first treats the task as a *Generalised Travelling Salesman Problem* (GTSP) instance, while the second uses an Machine Learning (ML) approach based on Text Similarity. The authors proved the performance of the system using LC-QuAD v1.0 [22] and QALD-7 [24] datasets, obtaining good results. They used accuracy to assess the system. The main limitation of EARL is that it does not address questions with hidden relations, that is, relations that cannot be directly inferred from the text.

Pan et al. [18] proposed a new framework for Relation Linking, called EERL, which exploits entities contained in questions to support the task. They expand the set of relation candidates by using properties that are logically connected to the target entities. The framework takes as input a natural language question and a knowledge graph, and it is composed of five modules: Relation Keyword Extractor, Keyword-based Relation Expansion, Entity Linking, Entity-based Relation Expansion, Relation Ranking. The first component, Relation Keyword Extractor, permits extraction of relation phrases contained in the question. In the second module, Keyword-based Relation Expansion, the authors use back-

ground knowledge to get a list of associated relation phrases. The Entity Linking module extrapolates the entities included in the question. Entity-based Relation Expansion is the main module, which allows to obtain all relation candidates, by expanding explicit and implicit relations. Relation expansion is based on the following hypothesis "The relations in questions are properties of the entities occurring in the question or properties of the types of these entities". In the end, the last module, Relation Candidate Ranking, is useful to select the best relations from the candidates. The authors test the system on three datasets: QALD-5 [23], QALD-7, LC-QuAD v1.0. The main weakness of the system is the real effectiveness of the proposed hypothesis, in fact, they noticed that some of the questions' relations don't appear in the relation candidates.

Ahmad Sakor et al. [20] introduced a rule-based approach to address the problem of joint entity and relation linking within the short text, called *Falcon*. The system mainly appeals to the fundamental principles of English morphology, such as compounding, right-hand rules for headword identification, and also uses an extended knowledge base. The authors use the well-known *spaCy* [11] library to annotate the text with POS tag information and then create a list of candidate entities and relations. For relations, candidates are proposed based on the verbs found in the text. Finally, a ranking mechanism organizes the candidates. Falcon was tested using the QALD-7 and LC-QuAD v1.0 datasets, outperforming the compared systems like EARL. Falcon, like EARL, has problems with questions that contain implicit relations due to the nature of the approach.

An interesting approach is the Semantic LINkinG system (SLING) proposed by Mihindukulasooriya et al. [15] SLING is a distant supervision-based system that leverages semantic parsing such as Abstract Meaning Representation (AMR). The authors use distance supervised techniques to address the challenge of lack of training data. The system is divided into two main components: Question Metadata Generation and Relation Linking. The first component takes as input the question text and its AMR graph. The graph is converted into a set of intermediate AMR triples, and the query text is used to obtain additional knowledge base information by using Entity Linking tools, like BLINK. The second component works using four different modules for Relation Linking, two supervised and two unsupervised approaches. Finally, the system aggregates the results of the modules to obtain a final ranked list of relations. SLING was executed on the well-known QALD-7, QALD-9, Lc-QUAD v1.0 datasets, demonstrating better performance than the related systems. It is not clear from the paper the limitations of this approach, so we think it may suffer from the same problem as the previously discussed RL approaches since it depends on the semantic mapping of the text that does not contain a reference to the implicit relationships.

The Generative relation linking for question answering over knowledge bases (GenRL) approach by Rossiello et al. [19] is considered the state-of-the-art in this type of task. GenRL works by using a Seq2Seq model to generate a sequence of relations from the question text. This approach extends the question text by introducing information about the entities present and their linked re-

lations, providing additional context for training the baseline model. Another interesting point is that GenRL starts from a pre-trained transformer model, called *BART* [14] that is fine-tuned depending on the different datasets present in the literature. The system also includes a validation mechanism that helps to improve the results obtained by the neural network. During the evaluation, GenRL demonstrated better performance than related systems on the DBpedia and Wikidata [25] knowledge bases.

Our work is based on the results described in [1], which were extended with a deeper experimental analysis. *SeqRL* has some similarities with GenRL, like the usage of Seq2Seq models, but it was developed independently by using different tools. The main difference between the two approaches is that they use a pre-trained seq2seq model, BART, which is based on a transformer architecture, while we use an LSTM-based model trained from scratch.

Also, they extend their model with knowledge integration and validation mechanisms, and this has positively impacted their performance.

3 Preliminaries

3.1 Knowledge Bases and SPARQL

A knowledge base can be defined as a formal description of a domain of interest that is suitable to be managed by an engine reasoning about the facts modeled in the knowledge base itself, e.g., query existing knowledge or obtain new knowledge. A formal description of knowledge as a set of concepts within a domain and the relationships that hold between them is called *ontology* [8].

Ontologies allow the organization of information in such a way that is interpretable by both humans and machines. To enable such a description, it is necessary to formally specify some components such as individuals, classes, attributes, and relations as well as restrictions, rules, and axioms. As a result, ontologies do not only introduce a shared and reusable knowledge representation but can also be used to infer new knowledge about the domain. An ontology, together with a set of individual instances of the ontology classes, constitutes a *knowledge base* (**KB**) [17].

To create an Ontology, it is necessary to resort to a set of languages and technologies defined to this aim. The most common ontologies are defined by using the RDF, RDFS, and OWL languages, which are maintained by W3C [28] and constitute the de-facto standards for this type of task. In this paper, we adopt RDF as the ontology specification language of reference. With RDF, information is represented by a collection of $\langle subject - predicate - object \rangle$ triples, where the *predicate* establishes a binary relationship between *subject* and *object*. So, a knowledge base is composed of a set of triples known as RDF-graph [27].

The resources in a KB can be defined using URIs, allowing to reference non-local resources. This property permits the interaction among multiple KBs, making the accessible information grow considerably, not only in volume but also in the diversity of domains. To retrieve the information, given a knowledge

base, one has to resort to a proper query language like *SPARQL*, the de-facto standard for this type of task.

SPARQL is an SQL-like language to query RDF-graphs. The syntax and semantics of the language allow the user to query a KB by defining triples looking for a match with *subject-predicate-object* patterns within the graph [28]. The following is an example of a SPARQL query:

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT ?place WHERE { dbr:Barack_Obama dbo:birthPlace ?place  }
```

The *PREFIX* construct defines a namespace useful to disambiguate concepts with the same name. In the third line, *SELECT* returns a list containing the values of the variable *?place*, while the *WHERE* clause contains the triples to be matched against the RDF-graph. In the above example, *dbr:Barack_Obama* identifies a KB resource, and *dbo:birthPlace* is a predicate used to link the place of birth. Note that the identifiers that begin with "?" are considered variables. The query models the answer to the question "Where was Barack Obama born?". In addition to the *SELECT* clause, there are other ways to obtain the results:

- *CONSTRUCT* constructs and returns an RDF-graph by substituting variables in the query pattern.
- *DESCRIBE* provides an RDF-graph describing the resources that were found.
- *ASK* returns a boolean value indicating whether the query pattern matches or not.

Like SQL, the language provides different constructs to modify the results of a query, as *ORDER BY*, *DISTINCT*, *OFFSET*, and *LIMIT*. Another significant construct is the FILTER clause, which permits to apply restrictions to results, by using, for instance, a simple boolean expression or a regular expression.

3.2 Recurrent Neural Networks

The approach proposed in this paper takes advantage of the development achieved in the Deep Learning field and addresses the current problem using Artificial Neural Networks (ANN) [6], specifically Recurrent Neural Networks (RNN) [7]. Due to a particular way of processing data, RNNs have contributed to obtaining good results in tasks involving the processing of data sequences, and therefore they are also good at processing natural language as well, if we consider the natural language as a semantically ordered sequence of words.

An RNN works by iterating over the elements of the sequence S and keeping a state h that contains information relative to what was already processed so that the result of processing the element at time t is also conditioned by the previous information $t-1$ [6]. At each time step t the state $h_{(t)}$ is updated by mean of $h_{(t)} = f(h_{(t-1)}, x_t)$, where f is a non-linear function. This way of operation is useful to capture semantic relationships between the words in a sentence.

Long short-term memory networks. When processing natural language, it is common to deal with sentences long enough to cause the memory mechanism employed by RNNs to collapse, making it hard to learn long-term dependencies. The phenomenon explained before is known as Vanishing Gradient Descent, and it was addressed by Hochreiter et. al [10] in 1997, introducing a particular type of RNN called *Long short-term memory* (LSTM). LSTM adds a way of transporting information through many time steps. Imagine a conveyor belt running parallel to the sequence you are processing. Information from the sequence can jump onto the conveyor belt at any point, be transported to a later timestep, and jump off, intact, when you need it. Essentially, an LSTM saves information for later, thus preventing older signals from gradually vanishing during processing [6]. The success of the LSTMs lies in a mechanism called Gates used to compute the hidden states. The gating mechanism can regulate the flow of information and decide what information is important to keep or throw away. This is done by mean of:

$$
\begin{aligned}
i &= \sigma(x_t U^i + s_{t-1} W^i) & f &= \sigma(x_t U^f + s_{t-1} W^f) \\
o &= \sigma(x_t U^o + s_{t-1} W^o) & g &= \tanh(x_t U^g + s_{t-1} W^g) \\
c_t &= c_{t-1} \circ f + g \circ i & s_t &= \tanh(c_t) \circ o
\end{aligned}
$$

where the input i, forget f, and output o represent gates that are squashed by the sigmoid into vectors of values between 0 and 1. Multiplying the vectors determines how much of the other vectors to let into the current input state. g is a candidate hidden state that is computed based on the current input and the previous hidden state. c_t is used as the internal memory, which is a combination of the previous memory c_{t-1} multiplied by the input gate, and the hidden state s_t is a combination of the internal memory and the output gate.

4 Neural network for Relation Linking

A naive neural approach for RL is to train a multi-class multi-label classifier that predicts predicates contained in the questions. As previously mentioned, the number of predicates is huge, just think that *DBpedia* contains more than 50.000 predicates, an unmanageable quantity in terms of resources for a classical neural network. In this paper, we tackle the task of RL as a *sequence to sequence* problem by using well-known Seq2Seq models. Therefore, the task of our system is to take a natural language question as input and returns a sequence of predicates contained in the underlying KB. For these models, the definition of input and output vocabularies is necessary. In this case, input vocabulary is the English language, while output vocabulary is the set of predicates included in DBpedia. All predicates have been obtained through a SPARQL query.

To provide the model with the proper input to improve the learning phase, we decided to rewrite the questions and the predicates to lowercase. We also normalized the predicates by deleting the prefix, to simplify the linking process.

The normalization process has been done because, in the vocabulary, there exist relations semantically equivalent that the model is not able to distinguish, like "dbo:name" and "dbp:name".

The output vocabulary has a reasonable number of predicates, especially when compared to the input one; for instance, consider that the Oxford English Dictionary contains more than 700.000 words. Neural networks, generally, can manage words, which they have already seen during the training phase and so contained in the training set, but currently, for this task, there are no datasets complete enough to include all words. To address this problem, we used pre-trained *Word Embedding*. In particular, we used models that are trained on *Wikipedia* through *FastText* [2]. They allow to have access to thousands of *word vectors*, learned over millions of words, and so, they can provide a vectorial representation even for terms not included in the training set. Note that output vocabulary is even primarily English, therefore, we used *FastText* models to extract the embeddings from predicates.

In this case, the *Seq2Seq* model has a classical structure Encoder-Decoder, in which the encoder has the goal to extract a vector V from the question, namely *context vector*, that encapsulates the entire meaning of the phrase; while the decoder tries to translate this vector in a sequence of predicates, belonging to output vocabulary. Both are composed of a *recurrent layer* of type *LSTM* with 128 units. During the training phase, we used the Teacher forcing, a famous training strategy for the recurrent network, which permits a rapid convergence. It consists in using as input of the decoder the *ground truths* of the previous step, instead of the last output predicted by the model. Figure 1 shows the model architecture used during the training phase.

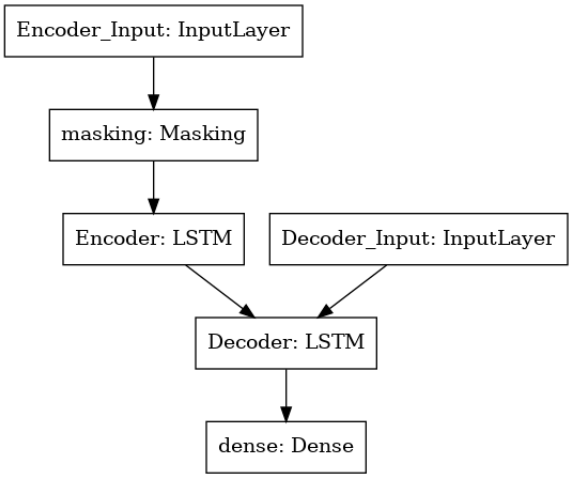
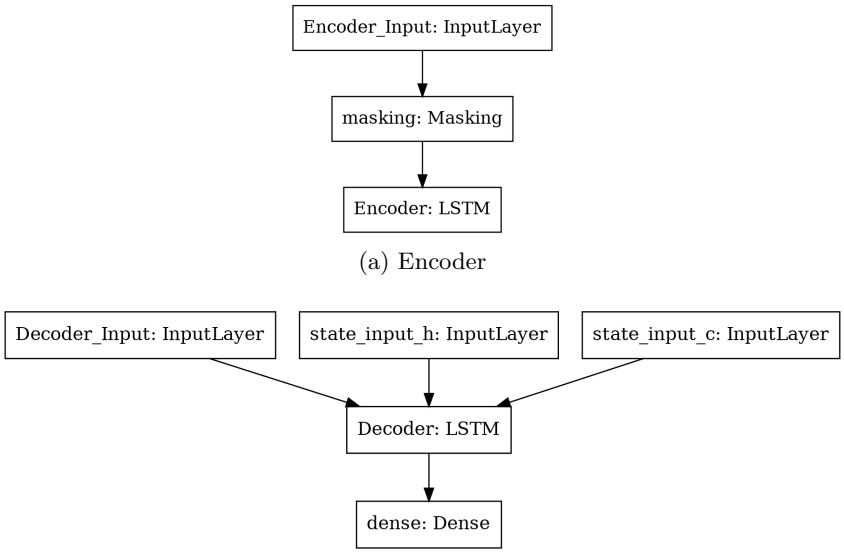


Fig. 1: Model architecture used during the training phase.

During the inference phase, the ground truths are not available, so the input of the decoder is simply what it predicts at the previous step. Also, in this phase, the iterative process of the recurrent network needs to be implemented at hand, splitting the original model into two sub-models: encoder and decoder, as shown in Figure 2.



(a) Encoder

(b) Decoder

Fig. 2: Model architectures used during the inference phase.

Another characteristic of *SeqRL*, suggested by the authors of article [21], is to reverse the words of the input question.

Figure 3 shows an example of an instance during the training phase. More deeply, the word embeddings are the actual input of the encoder and the de-

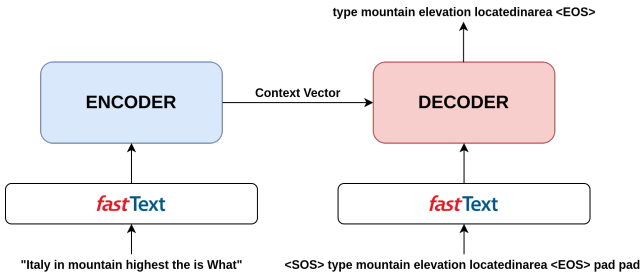


Fig. 3: Example during the training phase.

coder, generated by *FastText*; while the output of the decoder is a probability distribution over the output vocabulary, obtained through the *Softmax activation function*.

5 Experiments

5.1 Experiment Setup

We have implemented our models by using Keras, a well-known framework for *deep learning*, on top of *TensorFlow*. The experiments were executed on *Google Collaboratory*, a virtual environment based on *Jupyter Notebook*, and on a Linux server with 500 GB of ram and two *GPU Nvidia Tesla V100* with 16 GB of memory. We considered two well-known datasets for *KBQA* over the *DBpedia* ontology: *DBNQA* and *QALD-9*. To assess the system, we adopted precision, recall and F1-score metrics.

5.2 Evaluation on DBNQA

DBpedia Neural Question Answering (DBNQA) is a huge dataset for *KBQA*, composed by 894.499 question-query pairs. From each query, we extrapolated all predicates, so they can be used as ground truths. Also, we removed all statistical predicates, e.g. *wikiPageLength*. To evaluate the performance of the system, we decided to use *k-fold cross-validation*, with $k = 5$, since there are no other systems, to the best of our knowledge, that use DBNQA dataset for evaluation for Relation Linking, and so we cannot make a proper comparison. Notice that the aforementioned dataset is used mainly for Question Answering. For each step, the model was trained for 4 epochs, and Table 1 shows the obtained results. As shown, our system performs very well, reaching a score greater than 0.98 in all metrics. More in detail, we report that the dataset is quite repetitive, and this facilitates the behavior of the model.

In Table 2, we show a set of examples to analyze the behavior of the model. The examples in rows 1 and 7 show the ability of *SeqRL* to infer relations that are not directly expressed in the text, in fact, it is able to predict the unseen relations *architect* and *locatedinarea*.

Table 1: Results on *DBNQA*

	Precision	**Recall**	**F1**
Fold 1	0.9877	0.9877	0.9877
Fold 2	0.9866	0.9862	0.9864
Fold 3	0.9884	0.9882	0.9883
Fold 4	0.9883	0.9880	0.9882
Fold 5	0.9882	0.9879	0.9881
Final Average	**0.9878**	**0.9876**	**0.9877**

Table 2: Prediction examples

Question	Actual predicates	Predicted
Who built the Eiffel Tower?	dbo/dbp:architect	architect
Which instruments does Cat Stevens play?	dbo/dbp:instrument	type, instrument
Which book has the most pages?	rdf:type, dbo:Book, dbo:numberOfPages	type, book, numberofpages
When did princess Diana die?	dbp:deathDate	deathdate
Where did princess Diana die?	dbp:deathPlace	deathplace
What is the highest mountain in Italy?	dbo:locatedInArea, rdf:type, dbo:Mountain, dbo:elevation,	locatedinarea, type, mountain, elevation

Many approaches in the literature are not able to tackle this situation. For instance, Falcon[1], given the natural language question *"Who built the Eiffel Tower?"*, returns the wrong relation *numberBuilt*.

5.3 Evaluation on QALD-9

Our approach is designed mainly for huge datasets, like DBNQA. But, none of the related works use DBNQA to evaluate their system, so we decided to consider a dataset largely used in the literature to make comparison: QALD-9. The Question Answering over Linked Data (QALD) is a series of challenges that aim to provide benchmarks for assessing and comparing KBQA systems on DBpedia. We considered the benchmark proposed as part of the ninth edition of QALD, known as QALD-9. The dataset contains 558 question-query pairs in 11 different languages. The data is split into 408 training and 150 testing questions, and we focus on the ones expressed in English. It is important to note that this dataset is very challenging to be approached using learning techniques, given the very small training set not covering all questions types of the test set. For this reason, we trained our model on both DBNQA and QALD-9 training set. In particular we first, obtained a pre-trained model on DBNQA and then fine-tuned it performing training on QALD-9. Notice that DBNQA is built by using Lc-QUAD v1.0 and QALD-7-Train.

Table 3 shows comparison of *SeqRL* with some related approaches. The results for the related approaches have been taken from [19] (page 7 table 1). It is important to observe that the performance of FALCON 1.0, SLING and GenRL can be influenced by the problem of the equivalent predicates, due to the prefix that we ignore. As we can see, our approach performs good, outperforming systems like FALCON 1.0 and SLING, while the most recent work, GenRL, has a comparable performance (if not better performance).

[1] `https://labs.tib.eu/falcon/`

Table 3: Comparison on *QALD-9*

	Precision	Recall	F1
Falcon 1.0	0.23	0.23	0.23
SLING	0.39	0.50	0.44
GenRL	0.49	0.61	0.53
SeqRL	0.53	0.48	0.50

6 Conclusions and Future Work

The paper presented an approach based on deep neural networks for Relation Linking. We took advantage of the Sequence to Sequence models based on LSTM networks to address the problem by considering the predicates as a sequence of words. Our system demonstrated its great potential, obtaining optimal results on DBNQA and good ones on the challenging dataset QALD-9.

As future work, we want to extend our approach with other Natural Language Processing tools, like Entity Linking (EL). EL permits to identify entities into the question, which can be helpful to extract a set of candidate predicates to be used to simplify the process of linking. We also plan to extend our experiments considering other recently-developed KBQA benchmarks.

References

1. Barbara, V., Borroto, M., Ricca, F.: A Sequence to Sequence approach for Knowledge Base Relation Linking. Master's thesis, University of Calabria (Sep 2021)
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606 (2016)
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 **abs/2111.03000** (2021)
4. Cabrio, E., Cojan, J., Aprosio, A.P., Magnini, B., Lavelli, A., Gandon, F.: Qakis: an open domain qa system based on relational patterns. In: International Semantic Web Conference, ISWC 2012 (2012)
5. Dubey, M., Banerjee, D., Chaudhuri, D., Lehmann, J.: Earl: joint entity and relation linking for question answering over knowledge graphs. In: International Semantic Web Conference. pp. 108–126. Springer (2018)
6. Francois, C.: Deep learning with Python. Manning Publications Company (2017)
7. Giles, C.L., Kuhn, G.M., Williams, R.J.: Dynamic recurrent neural networks: Theory and applications. IEEE Trans. on Neur.Net. **5**(2), 153–156 (1994)
8. 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)
9. Hartmann, A., Marx, E., Soru, T.: Generating a large dataset for neural question answering over the DBpedia knowledge base (2018)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)

11. Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A.: spaCy: Industrial-strength Natural Language Processing in Python (2020). https://doi.org/10.5281/zenodo.1212303, https://doi.org/10.5281/zenodo.1212303
12. Kapanipathi, et al.: Question answering over knowledge bases by leveraging semantic parsing and neuro-symbolic reasoning. arXiv preprint arXiv:2012.01707 (2020)
13. 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)
14. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461 (2019)
15. Mihindukulasooriya, N., Rossiello, G., Kapanipathi, P., Abdelaziz, I., Ravishankar, S., Yu, M., Gliozzo, A., Roukos, S., Gray, A.: Leveraging semantic parsing for relation linking over knowledge bases. In: International Semantic Web Conference. pp. 402–419. Springer (2020)
16. Ngomo, N.: 9th challenge on question answering over linked data (qald-9). language **7**(1) (2018)
17. Noy, N.F., McGuinness, D.L., et al.: Ontology development 101: A guide to creating your first ontology (2001)
18. Pan, J.Z., Zhang, M., Singh, K., Harmelen, F.v., Gu, J., Zhang, Z.: Entity enabled relation linking. In: Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I., Hogan, A., Song, J., Lefrançois, M., Gandon, F. (eds.) The Semantic Web – ISWC 2019. pp. 523–538. Springer International Publishing, Cham (2019)
19. Rossiello, G., Mihindukulasooriya, N., Abdelaziz, I., Bornea, M., Gliozzo, A., Naseem, T., Kapanipathi, P.: Generative relation linking for question answering over knowledge bases. In: International Semantic Web Conference. pp. 321–337. Springer (2021)
20. Sakor, A., Onando Mulang', I., Singh, K., Shekarpour, S., Esther Vidal, M., Lehmann, J., Auer, S.: Old is gold: Linguistic driven approach for entity and relation linking of short text. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 2336–2346. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1243, https://aclanthology.org/N19-1243
21. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS. pp. 3104–3112 (2014)
22. Trivedi, P., Maheshwari, G., Dubey, M., Lehmann, J.: Lc-quad: A corpus for complex question answering over knowledge graphs. In: International Semantic Web Conference. pp. 210–218. Springer (2017)
23. Unger, C., Forascu, C., López, V., Ngomo, A.C.N., Cabrio, E., Cimiano, P., Walter, S.: Question answering over linked data (qald-5). In: CLEF (2014)
24. Usbeck, R., Ngomo, A.C.N., Haarmann, B., Krithara, A., Röder, M., Napolitano, G.: 7th open challenge on question answering over linked data (qald-7). In: Dragoni, M., Solanki, M., Blomqvist, E. (eds.) Semantic Web Challenges. pp. 59–69. Springer International Publishing, Cham (2017)
25. Vrandečić, D., Krötzsch, M.: Wikidata: A free collaborative knowledgebase. Commun. ACM **57**(10), 7885 (Sep 2014). https://doi.org/10.1145/2629489, https://doi.org/10.1145/2629489

26. W3C: SPARQL 1.1 query language (2013), https://www.w3.org/TR/sparql11-query/
27. W3C: Resource description framework (RDF) (2014), https://www.w3.org/RDF/
28. W3C: Semantic web standards (2014), https://www.w3.org