UNDER REVIEW

RESCIENCE C

Replication / ML Reproducibility Challenge 2020

[Re] Improving Multi-hop Question Answering over Knowledge Graphs using Knowledge Base Embeddings

Jishnu Jaykumar P[1] and Ashish Sardana[1]
[1]NVIDIA, Bengaluru, India

Edited by
(Editor)

Received
01 November 2018

Published
–

DOI
–

1 Introduction

Knowledge is the key to question answering task. Knowledge Graph (KG) is a multi-relational graph consisting of entities as nodes and relations among them as typed edges. KGs can accommodate a wide variety of facts, making them one of the potential candidates for intelligent decision-making. Question Answering over KG (KGQA) task aims to answer natural language queries posed over the KG. Multi-hop KGQA is a trending topic and has gained traction from both academia and industry recently. Multi-hop KGQA task involves reasoning over multiple edges of the KG to arrive at the correct answer. Earlier works on KGs(e.g. [1], [2], [3], [4]) have some element of sparsity, i.e. they do not capture all the facts available in the real world. Recent research on multi-hop KGQA has attempted to reduce this sparsity with the help of relevant external textual resources that are not readily available. On the other side, KG embeddings have emerged as an effective tool to overcome the KG sparsity by predicting missing links in the KG. Although effective, KG embeddings have not been explored for the multi-hop KGQA task. [5] fills this gap with the proposed EmbedKGQA method. This work intends to reproduce and perform an ablation (removing relation matching module) as well as an extended study on EmbedKGQA[5]. EmbedKGQA claims to be the first of its kind to use KG embeddings for multi-hop KGQA and improves over other state-of-the-art (SOTA) baselines.

2 Scope of reproducibility

According to [5], using ComplEx [6] KG embeddings significantly improves Hits@1 for multi-hop KGQA task and it has been proved with the help of the results on MetaQA [7] and WebQSP [8] datasets. This reproducibility work tries to test this claim and conducts experiments as mentioned in table:{2,3} of the original paper. Section 4.1 contains the corresponding results which support the claim with some anomalies.

3 Methodology

The authors of the original paper have open-sourced the code along with the data and pre-trained ComplEx KG embedding models. We have used the same codebase (commit:5d8fdbd4) and customized it for our purposes. In addition to this, we have added

Correspondence should be addressed to Jishnu Jaykumar P (jishnu.jayakumar182@gmail.com)
The authors have declared that no competing interests exist.
Code is available at https://github.com/jishnujayakumar/MLRC2020-EmbedKGQA. – SWH swh:1:dir:c95bc4fec7023c258c7190975279b5baf6ef6725.
Open peer review is available at https://openreview.net/forum?id=VFAwCMdWY7.

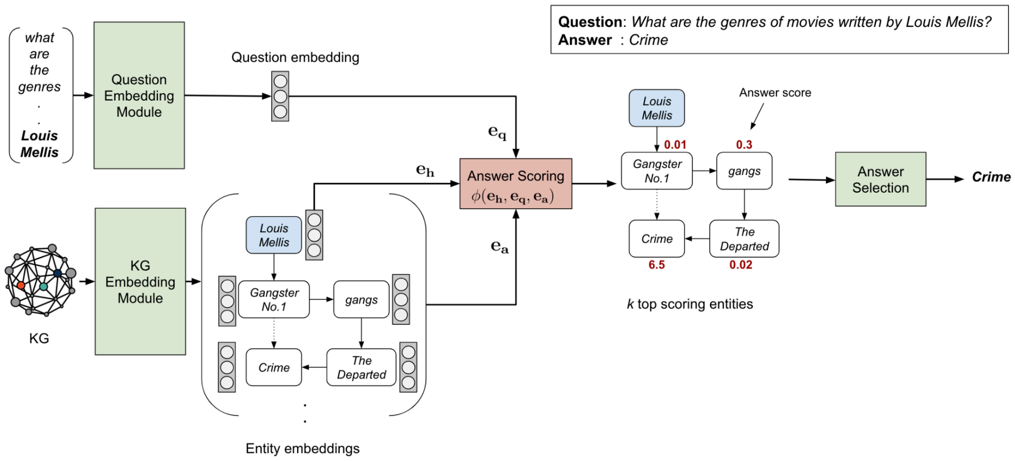

Figure 1. Overview of EmbedKGQA, the proposed method for Multi-hop KGQA. Image source: [5].

comprehensive documentation to make it more interpretable. Moreover, a command-line functionality is also added to easily configure various transformers models in the training workflow.

3.1 Model descriptions

As shown in Figure:1, EmbedKGQA has three modules:

1. KG Embedding Module - This module contains a KG embedding model called ComplEx [6] to learn embeddings for all entities in the input KG. 4 pretrained models have been shared by the author which contains 2 models for MetaQA-KG-{Full, 50} as well as 2 models WebQSP-KG-{Full,50} dataset. Details about the dataset are mentioned in section:3.2.
2. Question Embedding Module: Given a question q, head entity h and set of answer entities A, this module learns the question embeddings based on the score function defined by the KG embedding method used in 1.
3. Answer Selection Module: This module uses the outputs of module:1 and 2 to select the final answer by scoring the <head-entity, question> pair against all possible answers. The strategy is mentioned in section:{4.4, 4.4.1} of the original paper respectively.

3.2 Datasets

There are two datasets used in the original paper. MetaQA [7] and WebQSP [8]. Both datasets have two portions. (1) KG data (2) QA data. KG data for both are further divided into two categories. (1) Using the full KG (indicated by suffix KG-Full) and (2) Using only 50% of the facts in the respective KGs (indicated by suffix KG-50). The details of generating custom KG datasets are discussed here[1]. Both datasets are taken from here[2]. Statistics for table:{1, 2} have been taken from [5]. For generating question embeddings the question is placed between <s> and </s> tags for all transformers except Sentence-Transformer as it takes the input sentence in its pure form. The preprocessing used in the original code has been used here. No additional preprocessing has been performed from our end.

[1] https://github.com/malllabiisc/EmbedKGQA
[2] https://drive.google.com/drive/folders/1RlqGBMo45lTmWz9MUPTq-0KcjSd3ujxc (As of January, 2021)

Dataset	Train	Dev	Test
MetaQA 1-hop	96,106	9,992	9.947
MetaQA 2-hop	118,948	14,872	14,872
MetaQA 3-hop	114,196	14,274	14,274
WebQSP	2,998	100	1,639

Table 1. QA data statistics for each dataset according to [5]

Dataset	Triples	Entities	Relations	Experiment-Alias
MetaQA-KG-Full	135k	43k	9	MetaQA_full
WebQSP-KG-Full	5.7 million	1.8 million	γ	fbwq_full
MetaQA-KG-50	ϕ	-	-	MetaQA_half
WebQSP-KG-50	ψ	-	-	fbwq_half

Table 2. KG data statistics for each dataset. Refer[1] for more details.
Experiment-Alias is the name used for the respective datasets in experiments.
γ = Contains all facts that are within 2-hops of any entity mentioned in the questions of WebQSP.
ϕ = Contains only 50% of the triples (randomly selected without replacement).
ψ = Contains 50% of the edges sampled randomly from fbwq_full.

3.3 Hyper-Parameters

Hyper-parameters used to train the models aren't explicitly shared in the codebase or the paper, hence we decided to use the default values provided in codebase[1] to compensate for the lack of time. For reproduction, a pretrained model shared along with the data was used; ComplEx[6] was used as the knowledge graph embedding method for all the KG types, i.e., full and half of both datasets types. For reproducibility, hyper-parameters for training MetaQA and WebQSP QA models have been taken from section:{MetaQA[3], WebQuestionsSP[4]} of the original codebase respectively. For RoBERTa [9], a pretrained model **roberta-base** has been taken from HuggingFace transformers package [10]. Other hyper-parameters are populated by default values in codebase[1].

3.4 Experimental setup and code

Experiments have been performed on the NVIDIA DGX-1 server with 8xV100 GPUs, out of which 6 were used in this work. The metric used for validating the claims is Hits@1. According to [11], Hits@k is the proportion of test triples ranking in the top-k results. The code for this work is open-sourced on GitHub[5]. In addition to this, we have shared a couple of Docker images[6] for easy kick-starting of experiments without the hassle of setting up the environment. Following trained models are made available in our Docker image[6], chosen based on better performance in our extended study.

- TuckER KG embedding model for Meta-QA-{Full, 50}
- QA models trained using ComplEx as KG embedding model and SBERT mentioned in table:4 as question embedding model for WebQSP-KG-{Full, 50}

3.5 Computational requirements

This work has been performed on 6 V100-16GB GPUs connected via NVLink. NVLink reduced multi-GPU training time by 1/4. The time required for various reproductions

[3] https://github.com/malllabiisc/EmbedKGQA#metaqa
[4] https://github.com/malllabiisc/EmbedKGQA#webquestionssp
[5] https://github.com/jishnujayakumar/MLRC2020-EmbedKGQA
[6] https://github.com/jishnujayakumar/MLRC2020-EmbedKGQA#helpful-pointers

Type	MetaQA-KG-Full			MetaQA-KG-50		
	1-hop	2-hop	3-hop	1-hop	2-hop	3-hop
Train (t)	350 seconds	380 seconds	380 seconds	280 seconds	330 seconds	320 seconds
Validation (v)	42 seconds	95 seconds	147 seconds	47 seconds	108 seconds	182 seconds
T	10.89 hours	13.19 hours	14.64 hours	9.08 hours	12.16 hours	13.94 hours

Type	WebQSP-KG-Full	WebQSP-KG-50
Train (t)	280 seconds	300 seconds
Validation (v)	95 seconds	105 seconds
T	10.42 hours	10.92 hours

Table 3. Time for training/validation. Refer section:3.3 for hyper-parameters. (r=1, total_epochs=100)

For table:3, validate_every = The number of train routines before validation for a single epoch
Total runs (r) = Number of times the training has been performed for a particular task
Total train time (GPU hours) excluding early stopping, T = $(total\_epochs \times (t + v)) \times r$

Transformer	Pretrained-Model
RoBERTa	roberta-base
XLNet	xlnet-base-cased
ALBERT	albert-base-v2
SentenceTransformer (SBERT)	sentence-transformers/bert-base-nli-mean-tokens
Longformer	allenai/longformer-base-4096

Table 4. Pretrained models from HuggingFace transformers package [10]

are mentioned in table:{3}.

3.6 Extended Experiments

Apart from reproducing the results mentioned in the original paper, a couple of extended experiments have been performed to find answers to the following two questions:

1. Can recent KG embedding methods like TuckER [12] give higher accuracy on higher levels of hops, i.e., 3-hop scenario to be specific compared to [6] used in the original paper?
2. Can other transformer architectures like ALBERT [13], XLNet [14], Longformer [15] and SBERT [16]) improve the results on WebQSP [8]?

Details of hyper-parameters used for these experiments are available in our GitHub repository[5]. Various transformer models used for experiment-2 are mentioned in table:4.

4 Results

We report results for reproducibility as well as our extended experiments. The results of reproduction have a mixed nature while the ones for our extended experiments show

Model	RM	MetaQA-KG-Full			MetaQA-KG-50		
		1-hop	2-hop	3-hop	1-hop	2-hop	3-hop
EmbedKGQA	✔	**97.5**	**98.8**	**94.8**	**83.9**	**91.8**	70.3
EmbedKGQA (Reproduced)	✘	95.4	96.4	72.3	83.2	91.6	**71.2**
Δ	-	-2.1	-2.4	-22.5	-0.7	-0.2	**0.9**

Table 5. Hits@1 results for original and reproduced experiments using MetaQA-KG-{Full, 50} datasets. Δ = (Reproduced Hits@1 without RM) - (Original Hits@1 with RM)

Model	RM	WebQSP-KG-Full	WebQSP-KG-50
EmbedKGQA	✔	**66.6**	**53.2**
EmbedKGQA	✘	48.1	47.4
EmbedKGQA (Reproduced)	✘	54.9	41.3
$\Delta_{original}$	-	18.5	5.8
Δ	-	6.8	-6.1

Table 6. Hits@1 results for original and reproduced experiments using WebQSP-KG-{Full, 50} datasets. Δ=(Reproduced Hits@1 without RM) - (Original Hits@1 without RM), $\Delta_{original}$ = (Original Hits@1 with RM) - (Original Hits@1 without RM).

For table:{5, 6}, KG-Embedding-Model=ComplEx. RM=Relation Matching, ✔= inclusion, ✘= exclusion. The original values for EmbedKGQA are taken from [5]. Underline indicates anomaly due to the absence of RM module.

positive signs to support claim-1, 2. Detailed discussion about the results can be found in section:5. For all tables in this report, bold values indicate better performance.

4.1 Results reproducing original paper

We perform two experiments based on the two datasets introduced in section:3.2. These experiments provide vital information about the results mentioned in table:{2,5} of the original paper. The results of the two are reported in table:{5, 6} respectively. From the results of table:5 in [5] and table:6 in this report, it is evident that relation matching(RM) is an important component in multi-hop KGQA when the given KG is considerably large, i.e. {MetaQA, WebQSP} KG-Full; Definitely, WebQSP-KG-50 also shows improvement in presence of RM but the performance significantly improves when applied to KG-Full setting. The author of [5] had also expressed the same opinion in one of the virtual meetings. For table:{7, 8}, Δ= (TuckER Hits@k) - (ComplEx Hits@k).

4.2 Results beyond the original paper

We have conducted two additional experiments from our end to find an answer to claim:{1,2}. The results in table:{7,8} support claim-1 but with a caveat. On the other hand, values in table:9 improve upon the results reported by the original paper creating a new SOTA baseline. Additional experiments ingest custom hyper-parameters mentioned in our codebase in absence of the original hyper-parameters. None of these experiments include the RM module.

KG-Model	MetaQA-KG-Full								
	1-hop			2-hop			3-hop		
	Hits@1	Hits@5	Hits@10	Hits@1	Hits@5	Hits@10	Hits@1	Hits@5	Hits@10
ComplEx	95.39	**99.83**	99.97	**96.46**	**99.02**	99.27	72.33	93.27	95.66
TuckER	**95.51**	99.81	99.97	93.13	98.7	**99.28**	**73.81**	**93.6**	**96.09**
Δ	**0.12**	−0.02	0	−3.33	−0.32	**0.01**	**1.48**	**0.33**	**0.43**

Table 7. Comparison of ComplEx with TuckER based on Hits@k results for MetaQA-KG-Full dataset. $k \in \{1, 5, 10\}$.

KG-Model	MetaQA-KG-50								
	1-hop			2-hop			3-hop		
	Hits@1	Hits@5	Hits@10	Hits@1	Hits@5	Hits@10	Hits@1	Hits@5	Hits@10
ComplEx	**83.24**	**89.83**	**91.22**	**91.63**	**97.08**	**98.04**	71.2	90.77	93.72
TuckER	83	89.36	90.41	86.07	94.66	96.4	**71.96**	**91.16**	**93.94**
Δ	−0.24	−0.47	−0.81	−5.56	−2.42	−1.64	**0.76**	**0.39**	**0.22**

Table 8. Comparison of ComplEx with TuckER based on Hits@k results for MetaQA-KG-50 dataset. $k \in \{1, 5, 10\}$.

Model	WebQSP-KG-Full			WebQSP-KG-50		
	Hits@1	Hits@5	Hits@10	Hits@1	Hits@5	Hits@10
RoBERTa [9]	54.96	67.62	71.97	41.27	51.14	54.19
XLNet [14]	51.98	64.44	69.11	39.33	49.25	52.04
ALBERT [13]	47.31	59.83	63.98	31.15	42.31	45.68
Longformer [15]	54.9	66.77	70.47	41.92	51.98	54.83
SBERT [16]	**55.55**	**68.98**	**72.74**	**44.65**	**53.86**	**56.13**
Δ	**0.59**	**1.36**	**0.77**	**3.38**	**2.72**	**1.94**

Table 9. Hits@k results for recent transformer models by [10] used for generating question embeddings. KG-Embedding-Method=ComplEx, Δ= (SBERT_Hits@k - RoBERTa_Hits@k), $k \in \{1, 5, 10\}$

5 Discussion

The reproducibility results from table:{5,6} corroborate the claims mentioned in section:2 to some extent. Reproduced version is within the ± 2.4 range (positive value indicates better performance and vice-versa) except for the MetaQA-KG-Full dataset's 3-hop and WebQSP-KG-Full scenario which has a significant drop of 22.5% and 18.5% respectively. The absence of RM module has been reported and discussed here[7,8,9]. For a given question, the RM module uses it's context to extract useful information from the available edges present in the KG. This information is further plugged into the answer selection module to select more relevant answers. Thus, relation matching is a vital component in multi-hop question answering, especially in the KG-Full setting where more edges are present w.r.t. KG-50 setting or any smaller KG w.r.t. the KG-Full setting. Results from table:{5,6} corroborates the previous statement. Moreover, MetaQA-KG-50 3-hop outperforms the original model by a margin of +0.9% without using RM which is an interesting observation. Apart from one reported anomaly, the reproduced results are pretty close to the original results in the case of the MetaQA dataset. The default set of hyper-parameters mentioned in the original codebase(Refer section: 3.3) were used in the reproducibility study. The anomaly in WebQSP-KG-Full,i.e. 18.5% drop bolsters the importance of RM in the KG-Full setting. The reproduced results for WebQSP-KG-50 are within the ± 7% range. The use of different hyper-parameters can be one of the possible answers to this variation. This value is significant but not w.r.t. WebQSP-KG-Full's drop of 18.5% which again strengthens the importance of RM in the KG-Full setting. As mentioned in 4.1, RM is highly useful when the KG is considerably large. From table:{7, 8}, it is clear that TuckER [12] performs better than ComplEx [6] for the 3-hop scenario for both MetaQA-KG datasets, i.e., Full and 50. Though these results strengthen claim-1, a more comprehensive set of tests may lead to a concrete conclusion (e.g., experiments employing a broader set of hyper-parameters). According to table:9, in all the cases, SBERT [16] outperforms RoBERTa [9] used in the original paper creating a new SOTA benchmark which supports claim-2. Some experiments didn't work out due to the lack of time. E.g. Using RelationalTucker3 [17] and SimplE [18] to test claim-1. Furthermore, the hyper-parameter search couldn't be done due to the same reason hence we had to pick the default ones mentioned in the codebase. All these create room for further experiments and improvements.

5.1 What was easy

The paper was straightforward to understand. The open-sourced codebase helped us get kick-started.

5.2 What was difficult

The structure of the codebase made it difficult to navigate it. Since the code relied upon different techniques for the two datasets, the development of one function that trains different kinds of KG embeddings and another function that trains different kinds of QA models for both datasets was difficult. MetaQA uses LSTM/GRU [19] / [20] while WebQSP uses RoBERTa [9] to perform the same task of generating question embeddings. Also, training KG embeddings for MetaQA yields files in the form of NumPy [21] files while WebQSP uses LibKGE [22] for the same purpose which produces LibKGE specific KG embedding(KGE) models. Reproduction and the extensive study were a bit hard in the beginning as KGE and question embedding methodology varied for both datasets. After having a couple of virtual meetings with the author and code review, it became easier to

[7] https://github.com/malllabiisc/EmbedKGQA/issues/1
[8] https://github.com/malllabiisc/EmbedKGQA/issues/51
[9] https://github.com/malllabiisc/EmbedKGQA/issues/56

conduct the planned experiments. The unavailability of hyper-parameters used to train each module increased the experiment cycle multi-fold.

5.3 Communication with original authors

We had a couple of virtual meetings with the primary author of [5]. Though it was daunting to understand the codebase due to the reasons mentioned in section:5.2 with the help and support of the author, it became easier to navigate the codebase.

5.4 Future Scope

We think that there is a wide range of empirical analysis and experimentation that can be performed for multi-hop QA task, out of which we are sharing a few here:

1. Using KG embedding compression techniques like [23] in KG Embedding Module.
2. Using recent transformer models like Performer [24], Reformer [25] etc. for generating question embeddings.
3. Using low-dimensional hyperbolic KG embeddings [26] in KG embedding module along with hyperbolic word embeddings [27] for question embedding module.
4. A new approach for sentence embedding, SBERT-WK [28] instead of SBERT [16] can be tried out.

References

1. F. M. Suchanek, G. Kasneci, and G. Weikum. "Yago: A Core of Semantic Knowledge." In: **Proceedings of the 16th International Conference on World Wide Web**. WWW '07. Banff, Alberta, Canada: Association for Computing Machinery, 2007, pp. 697–706. DOI: 10.1145/1242572.1242667. URL: https://doi.org/10.1145/1242572.1242667.
2. Google. **Freebase Data Dumps**. 2013. URL: https://developers.google.com/freebase/data.
3. J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al. "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia." In: **Semantic web** 6.2 (2015), pp. 167–195.
4. T. Mitchell et al. "Never-Ending Learning." In: **Commun. ACM** 61.5 (Apr. 2018), pp. 103–115. DOI: 10.1145/3191513. URL: https://doi.org/10.1145/3191513.
5. A. Saxena, A. Tripathi, and P. Talukdar. "Improving Multi-hop Question Answering over Knowledge Graphs using Knowledge Base Embeddings." In: **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**. Online: Association for Computational Linguistics, July 2020, pp. 4498–4507. DOI: 10.18653/v1/2020.acl-main.412. URL: https://www.aclweb.org/anthology/2020.acl-main.412.
6. T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. "Complex Embeddings for Simple Link Prediction." In: **Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48**. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 2071–2080.
7. Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song. "Variational Reasoning for Question Answering with Knowledge Graph." In: **AAAI**. 2018.
8. W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh. "The Value of Semantic Parse Labeling for Knowledge Base Question Answering." In: **Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 201–206. DOI: 10.18653/v1/P16-2033. URL: https://www.aclweb.org/anthology/P16-2033.
9. Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. **RoBERTa: A Robustly Optimized BERT Pretraining Approach**. 2019. arXiv:1907.11692 [cs.CL].
10. T. Wolf et al. "Transformers: State-of-the-Art Natural Language Processing." In: **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations**. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.
11. Y. Wang, D. Ruffinelli, R. Gemulla, S. Broscheit, and C. Meilicke. **On Evaluating Embedding Models for Knowledge Base Completion**. 2019. arXiv:1810.07180 [cs.AI].

12. I. Balažević, C. Allen, and T. M. Hospedales. "Tucker: Tensor factorization for knowledge graph completion." In: **arXiv preprint arXiv:1901.09590** (2019).
13. Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. "Albert: A lite bert for self-supervised learning of language representations." In: **arXiv preprint arXiv:1909.11942** (2019).
14. Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. "Xlnet: Generalized autoregressive pretraining for language understanding." In: **Advances in neural information processing systems**. 2019, pp. 5753–5763.
15. I. Beltagy, M. E. Peters, and A. Cohan. **Longformer: The Long-Document Transformer**. 2020. arXiv:2004.05150 [cs.CL].
16. N. Reimers and I. Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." In: **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing**. Association for Computational Linguistics, Nov. 2019. URL: https://arxiv.org/abs/1908.10084.
17. Y. Wang, S. Broscheit, and R. Gemulla. **A Relational Tucker Decomposition for Multi-Relational Link Prediction**. 2019. arXiv:1902.00898 [cs.LG].
18. S. M. Kazemi and D. Poole. **SimplE Embedding for Link Prediction in Knowledge Graphs**. 2018. arXiv:1802.04868 [stat.ML].
19. S. Hochreiter and J. Schmidhuber. "Long short-term memory." In: **Neural computation** 9.8 (1997), pp. 1735–1780. URL: https://www.bibsonomy.org/bibtex/2a4a80026d24955b267cae636aa8abe4a/dallmann.
20. J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. **Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling**. 2014. arXiv:1412.3555 [cs.NE].
21. C. R. Harris et al. "Array programming with NumPy." In: **Nature** 585 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.
22. S. Broscheit, D. Ruffinelli, A. Kochsiek, P. Betz, and R. Gemulla. "LibKGE-A knowledge graph embedding library for reproducible research." In: **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations**. 2020, pp. 165–174.
23. M. Sachan. "Knowledge Graph Embedding Compression." In: **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**. 2020, pp. 2681–2691.
24. K. Choromanski et al. **Rethinking Attention with Performers**. 2020. arXiv:2009.14794 [cs.LG].
25. N. Kitaev, Ł. Kaiser, and A. Levskaya. **Reformer: The Efficient Transformer**. 2020. arXiv:2001.04451 [cs.LG].
26. I. Chami, A. Wolf, D.-C. Juan, F. Sala, S. Ravi, and C. Ré. "Low-Dimensional Hyperbolic Knowledge Graph Embeddings." In: **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**. Online: Association for Computational Linguistics, July 2020, pp. 6901–6914. DOI: 10.18653/v1/2020.acl-main.617. URL: https://www.aclweb.org/anthology/2020.acl-main.617.
27. B. Dhingra, C. Shallue, M. Norouzi, A. Dai, and G. Dahl. "Embedding Text in Hyperbolic Spaces." In: **Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)**. New Orleans, Louisiana, USA: Association for Computational Linguistics, June 2018, pp. 59–69. DOI: 10.18653/v1/W18-1708. URL: https://www.aclweb.org/anthology/W18-1708.
28. B. Wang and C. .-. J. Kuo. "SBERT-WK: A Sentence Embedding Method by Dissecting BERT-Based Word Models." In: **IEEE/ACM Transactions on Audio, Speech, and Language Processing** 28 (2020), pp. 2146–2157.