EmbedKGQA: Improving Multi-hop Question Answering over Knowledge Graphs using Knowledge Base Embeddings

Anonymous Author(s)
Affiliation
Address
email

Reproducibility Summary

Scope of Reproducibility

Our work consists of three parts: (1) Reproducing results from Saxena et al. [2020] (2) Adding more experiments by replacing the knowledge graph embedding method (3) and exploring the question embedding method using various transformer models. Based on the paper implementation shared by the authors, we have reproduced the results for the EmbedKGQA method. We do not perform relation matching as it was non-trivial to implement it inside the given codebase.

Methodology

We have used the code provided by Saxena et al. [2020] for reproducibility. In addition to making the codebase more modular and easy to navigate, we have made changes to incorporate different transformers in the question answering module. Question-Answering models were trained from scratch as no pre-trained models were available for our particular dataset. The code[1] for this work is open sourced.

Results

We were able to reproduce the accuracy(hits@1) to be within $\pm$**2.35%** of reported value (in most cases). The anomaly was observed in 3 cases. [1] In MetaQA-KG-Full(3-hop) dataset due to absence of relation matching. [2] In WebQSP-KG-50 and [3] WebQSP-KG-Full datasets due to absence of original hyper-parameters but the difference between the reproduced and the original results were $\approx$12% in both cases. From our experiments on the QA model, we've found another transformer architecture (sentence transformer) to produce even better accuracy than the paper. Replacing RoBERTa (Liu et al. [2019]) with SBERT (Reimers and Gurevych [2019]) increased the absolute accuracy by $\approx$3.4% in half KG case and $\approx$0.6% in the full KG case. (KG: Knowledge Graph, "$\approx$": Approximately)

What was easy

As the code was available, we didn't have to implement the paper which allowed us to focus on author's claim validation, perform extended experiments and explore models. In addition to this, pretrained KG embedding models were shared which helped in the reproduction experiment.

What was difficult

The codebase was missing comments defining functions/classes/attributes etc., which made it difficult to review the code and modify it. In addition to large training times for question answering models, the knowledge graph embeddings also required significant amount of compute resources.

Communication with original authors

We had a couple of virtual meetings with Apoorv Saxena, the primary author.

[1] https://github.com/jishnujayakumar/MLRC2020-EmbedKGQA

Submitted to ML Reproducibility Challenge 2020. Do not distribute.

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 it 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 lately. Multi-hop KGQA task involves reasoning over multiple edges of the KG to arrive at the correct answer.

Earlier works on KGs(e.g. Suchanek et al. [2007], Google [2013], Lehmann et al. [2015], Mitchell et al. [2018]) 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 which 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 used for the mulit-hop KGQA task and Saxena et al. [2020] fills this gap with the proposed EmbedKGQA method.

This work intends to reproduce and perform an ablation study on EmbedKGQA(Saxena et al.[2020]). EmbedKGQA claims to be the first of its kind to use KG embeddings for multi-hop KGQA and improve over other state-of-the-art baselines.

2 Scope of reproducibility

According to Saxena et al. [2020], using ComplEx(Trouillon et al. [2016]) KG embeddings significantly improves Hits@1 for multi-hop question answering task and it's been proved with the help of the results on MetaQA(Zhang et al. [2018]) and WebQSP(Yih et al. [2016]) datasets respectively. This reproducibility work tries to test this claim and conducts experiments as mentioned in the original paper. Corresponding results have been reported in section 4.1 which shows positive results as well as anomalous behaviour.

3 Methodology

The author's of the original paper have open-sourced the code along with the data and pre-trained ComplEx KG embedding model. We have used the same codebase, added comments to make it more interpretable and also added a command line functionality to easily configure the hyper-parameters of 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 (Trouillon et al. [2016]) 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 3.2.
2. Question Embedding Module: This module includes a transformer model called RoBERTa (Liu et al. [2019]) that learns an embedding for the question. Details of the model can be found here[4].
3. Answer Selection Module: This module uses the outputs of 1 and 2 selects the final answer by incorporating the question and relation similarity scores.

3.2 Datasets

There are two datasets used in the original paper. MetaQA (Zhang et al. [2018]) and WebQSP (Yih et al. [2016]). 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[2]. Both datasets are taken from here[3]. Statistics for table:1 and 2 have been taken from Saxena et al. [2020]. For generating question embeddings the question is placed between <s> and </s> tags for all transformers except SentenceTransformer as it takes the input

[2]https://github.com/malllabiisc/EmbedKGQA

[3]https://drive.google.com/drive/folders/1RlqGBMo45lTmWz9MUPTq-0KcjSd3ujxc

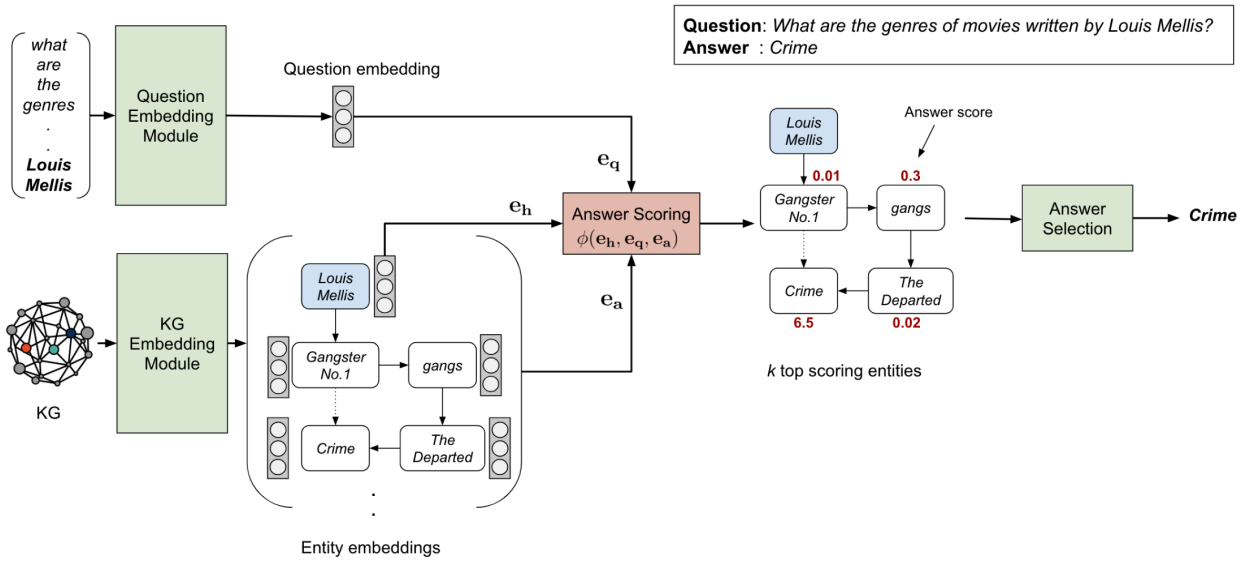

Figure 1: Overview of EmbedKGQA, the proposed method for Multi-hop QA over Knowledge Graphs (KGQA). Image is taken from Saxena et al. [2020].

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 Saxena et al. [2020]

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[2] 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.

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.

3.3 Hyper-Parameters

Hyper-parameters used to train the model aren't explicitly shared in the codebase or in the paper, hence we decided to use the default values provided in the codebase[2]. For the purpose of reproduction, a pretrained model shared along with the data was used. The ComplEx(Trouillon et al. [2016]) was used as the knowledge graph embedding method for all the KG types, i.e. full and half of both datasets types.

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. For, hyper-parameters refer section:3.3. (r=1, 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 = $(total_epochs \times (t + v)) \times r$

For the purpose of reproducibility, hyper-parameters for training MetaQA and WebQSP QA models have been taken from Section:MetaQA[4] and Section:WebQuestionsSP[5] of the original codebase respectively. For RoBERTa (Liu et al. [2019]), a pretrained model 'roberta-base' has been taken from HuggingFace transformers package (Wolf et al. [2020]). Other hyper-parameters are populated by default values in the codebase[2].

3.4 Experimental setup and code

The experiments has been performed on NVIDIA DGX-1 server with 6xV100 GPUs. The metric used for validating the claims is Hits@1. According to Wang et al. [2019b], Hits@K is the proportion of test triples ranking in the top-K results. The code for this report is hosted on Github[6].

Following trained models are made available in our docker image[7].

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

3.5 Computational requirements

8 V100 16GB GPUs were used for this work. GPUs are connected via NVLink which reduced multi-gpu training time by 1/4. The time required for various reproductions are mentioned in table:5,6

3.6 Extended Experiments

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

1. Can recent KG embedding methods like TuckER (Balažević et al. [2019]) give higher accuracy on higher levels of hops, i.e. 3-hop scenario to be specific compared to Trouillon et al. [2016] used in the original paper?
2. Can other transformer architectures like ALBERT (Lan et al. [2019]), XLNet (Yang et al. [2019]), Longformer (Beltagy et al. [2020]) and SBERT (Reimers and Gurevych [2019]) improve the results on WebQSP (Yih et al. [2016])?

Details of hyper-parameters used for these experiments are available on our Github repository[6]. Various transformer models used for experiment-2 are mentioned in table: 4

[4]https://github.com/malllabiisc/EmbedKGQA#metaqa
[5]https://github.com/malllabiisc/EmbedKGQA#webquestionssp
[6]https://github.com/jishnujayakumar/MLRC2020-EmbedKGQA
[7]https://hub.docker.com/r/jishnup/embed_kgqa

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 (Wolf et al. [2020])

Model	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. Bold values indicate better performance. Underline indicates the anomaly due to the absence of relation matching feature.

Model	WebQSP KG-Full	WebQSP KG-50
EmbedKGQA	**66.6**	**53.2**
EmbedKGQA (Reproduced)	54.9	41.3
Δ	-11.7	-11.9

Table 6: Original and reproduced stats for Hits@1 metric.

For table:{5, 6} and KG-Embedding-Model=ComplEx
Δ=(Reproduced Hits@1 value) - (Original Hits@1 value)

4 Results

We report results for both, the reproducibility and our extended experiments. The results of reproduction have a mixed nature.

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-2,3 of the original paper. The results of the two are reported in table-5 and 6 respectively. For making the most out of the result tables, the caption of the respective tables provide more insights about the results.

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 supports claim-1 but with a caveat. On the other hand table-9 values improve upon the results reported by the original paper and creating new SOTA baseline.

5 Discussion

The reproducibility results from table-5,6 corroborates the claims mentioned in Section:2 to some extent. Reproduced version in within ± 2.4 range (positive value indicates better performance and vice-versa except for MetaQA-KG-50 dataset's 3-hop scenario which has a significant drop of 22.5%. This happened due to the absence relation matching

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@10	Hits@1	Hits@5	Hits@5	Hits@10	Hits@1
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. $k \in \{1, 5, 10\}$.

For table-7, 8, Δ=(TuckER Hits@k value) - (ComplEx Hits@k value)

Question-Embedding-Method	WebQSP-KG-Full			WebQSP-KG-50		
	Hits@1	Hits@5	Hits@10	Hits@1	Hits@5	Hits@10
RoBERTa (Liu et al. [2019])	54.96	67.62	71.97	41.27	51.14	54.19
XLNet (Yang et al. [2019]	51.98	64.44	69.11	39.33	49.25	52.04
ALBERT (Lan et al. [2019])	47.31	59.83	63.98	31.15	42.31	45.68
Longformer (Beltagy et al. [2020])	54.9	66.77	70.47	41.92	51.98	54.83
SBERT (Reimers and Gurevych [2019]	**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 different transformer models Wolf et al. [2020] models used for question embedding generation. KG-Embedding-Method=ComplEx, Δ=SBERT_Hits@k value - RoBERTa_Hits@k value

module (reported here[8] and here[9]) and hence it can be concluded that relation matching is vital component in multi-hop question answering. Also,3-hop half kg outperforms the original model by a small margin of +0.9%.

From table-7, 8, it is clear that TuckER performs better than ComplEx for 3-hop scenario for both MetaQA-KG datasets, i.e. Full and 50. Though these results support the claim-1, more comprehensive set of experiments have to be performed for a strong conclusion.(e.g. experimentation using a broader set of hyper-parameters).

According to table-9, in all the cases, SBERT (Reimers and Gurevych [2019]) outperforms RoBERTa (Liu et al. [2019]) used in the original paper creating a new SOTA benchmark which supports the claim-2.

There were some experiments which didn't work out due to lack of time. E.g. Using RelationalTucker3 (Wang et al. [2019a]) and SimplE (Kazemi and Poole [2018]) for test claim-1)

5.1 What was easy

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

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

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 2 datasets, it was difficult to develop 1 function which trains different kinds of KG embeddings and 1 other function which trains different kinds of QA models for both datasets. MetaQA uses LSTM/GRU (Hochreiter and Schmidhuber [1997] / Chung et al. [2014] while WebQSP uses / RoBERTa (Liu et al. [2019]) to perform the same task of generating question embeddings. Also, training KG embeddings for metaqa yields files in the form of numpy files while WebQSP uses LibKGE (Broscheit et al. [2020]) for the same purpose which produces LibKGE specific KGE models. Since, KG embeddings and question embedding methodology differs for both datasets, reproduction and extensive study was a bit hard in the beginning but eventually after going through couple of meetings with the author and code review it became a normal routine.

The unavailability of hyper-parameters used to train each module increased the experiment cycle by multi-fold as each experiment took hours even on the powerful GPU server that we were using.

5.3 Communication with original authors

A couple of virtual meetings were organised with Apoorv, the first author. An extensive QA session was held. At the beginning, it was daunting to understand the code due to the reasons mentioned in Section:5.2 but with the help and support of the author it became easier to navigate the codebase.

6 Future Scope

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

1. KG embedding compression (Sachan [2020])
2. Using recent transformer models like PerformerChoromanski et al. [2020], Reformer (Kitaev et al. [2020]) etc. for generating question embeddings.
3. Using low-dimensional hyperbolic KG embeddings (Chami et al. [2020]) in KG embedding module along with hyperbolic word embeddings (Dhingra et al. [2018]) for question embedding module.
4. A new approach for sentence embedding, SBERT-WK (Wang and Kuo [2020]) instead of SBERT (Reimers and Gurevych [2019]) can be tried out.

References

Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*, 2019.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

Samuel Broscheit, Daniel Ruffinelli, Adrian Kochsiek, Patrick Betz, and Rainer 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*, pages 165–174, 2020.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6901–6914, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.617. URL `https://www.aclweb.org/anthology/2020.acl-main.617`.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2020.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

Bhuwan Dhingra, Christopher Shallue, Mohammad Norouzi, Andrew Dai, and George Dahl. Embedding text in hyperbolic spaces. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, pages 59–69, New Orleans, Louisiana, USA, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-1708. URL `https://www.aclweb.org/anthology/W18-1708`.

Google. Freebase data dumps. https://developers.google.com/freebase/data, 2013.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs, 2018.

Nikita Kitaev, ukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195, 2015.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-ending learning. *Commun. ACM*, 61(5):103115, April 2018. ISSN 0001-0782. doi: 10.1145/3191513. URL https://doi.org/10.1145/3191513.

Nils Reimers and Iryna 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, 11 2019. URL https://arxiv.org/abs/1908.10084.

Mrinmaya Sachan. Knowledge graph embedding compression. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2681–2691, 2020.

Apoorv Saxena, Aditay Tripathi, and Partha 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*, pages 4498–4507, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.412. URL https://www.aclweb.org/anthology/2020.acl-main.412.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 697706, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936547. doi: 10.1145/1242572.1242667. URL https://doi.org/10.1145/1242572.1242667.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 20712080. JMLR.org, 2016.

B. Wang and C. . J. Kuo. SBERT-WK: A sentence embedding method by dissecting BERT-based word models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2146–2157, 2020.

Yanjie Wang, Samuel Broscheit, and Rainer Gemulla. A relational tucker decomposition for multi-relational link prediction, 2019a.

Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, Samuel Broscheit, and Christian Meilicke. On evaluating embedding models for knowledge base completion, 2019b.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763, 2019.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina 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)*, pages 201–206, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-2033. URL `https://www.aclweb.org/anthology/P16-2033`.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. Variational reasoning for question answering with knowledge graph. In *AAAI*, 2018.