KRAM: Question Answering over Knowledge Graphs

Claudia PREDA
Computer Science & Engineering Department
Faculty Of Automatic Control And Computers
University Politehnica Of Bucharest
Bucharest, Romania
claudia.preda2307@stud.acs.upb.ro

Mihai PREDOIU
Computer Science & Engineering Department
Faculty Of Automatic Control And Computers
University Politehnica Of Bucharest
Bucharest, Romania
mihai.predoiu@stud.acs.upb.ro

Rares FOLEA
Computer Science & Engineering Department
Faculty Of Automatic Control And Computers
University Politehnica Of Bucharest
Bucharest, Romania
rares.folea@stud.acs.upb.ro

Andrei VATAVU
Computer Science & Engineering Department
Faculty Of Automatic Control And Computers
University Politehnica Of Bucharest
Bucharest, Romania
andrei.vatavu@stud.acs.upb.ro

***Abstract*—KRAM is a machine-learning based solution for Question Answering using Knowledge Graphs, mainly focused on creating and learning embeddings over arbitrary Knowledge Graph representations.**

***Index Terms*—KRAM, knowledge graph, question answering, machine learning, MetaQA, WebQuestionsSP, Complex WebQ, Embeddings**

CONTENTS

I. INTRODUCTION

This paper documents various academic solutions [10], [11], [19], [3], [2], [5], [16], [4] from the literature to the problem of **Question Answering over Knowledge Graphs**.

KRAM aims to become an efficient machine-learning solution that approaches the problem of **Question Answering using Knowledge Graphs,** being at the intersection of Machine Learning approaches, Semantics and Knowledge Graphs and Natural Language Processing. *The utmost goal is to create and learn embeddings over arbitrary Knowledge Graph representations.*

We could not agree more with the starting statement of [2]: "*The long-standing goal of natural language understanding is the development of systems which can acquire knowledge from text collections*". We truly believe that natural language processing should aim for more than strings, but for having a connotation. *Google* introduced their Knowledge Graph[1] as a solution to simple human needs: to learn and expand the horizons is at the heart of search: the result was proposing a model that understands real-world entities and their relationships to one another: **things, not strings**.

Knowledge Graphs are often incomplete, meaning that the direct relationship between two entities is missing. This relationship can be deducted from others al-

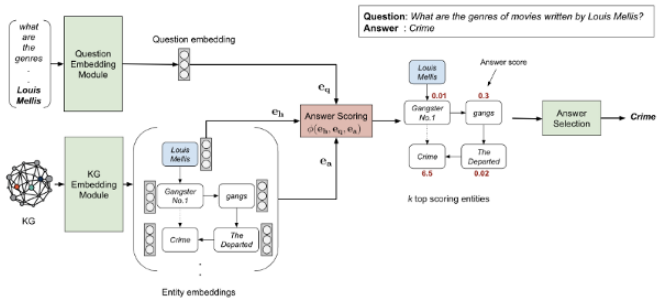

Figure 1: EmbedKGQA, the algorithm learns embeddings for the graph and the given NLP question. The embeddings are fed to the Answer Selection Module to select the appropriate answer based on relation similarity scores.

ready existing in the graph, however many of the algorithms proposed on this matter impose heuristic neighborhood limits that ignores the answer to a question. The missing links represents an additional challenge for **KGQA**, especially for *multi-hop reasoning*. This problem is addressed by introducing KG embeddings methods which perform prediction of the missing links, a highly approached subject of research in recent years. Given an incomplete **KG**, the task is to find what unknown links are valid. The following paper [10] proposes **EmbedKGQA**, a three modules-based solution that promotes the use of embeddings to improve multi-hop questions answering over knowledge graphs. The first module, KG Embedding Module, uses **ComplEx** [14], a method for embedding entities and relationships in KG. It represents them as vectors and relation matrices in a complex space. The *Question Embedding Module embeds the natural language questions* using **RoBERTa** [9]. The third module is called the Answer Selection Module and reduces the set of candidate answer entities by selecting the final answer, the best-scored for smaller **KGs** and for the larger ones, pruning strategy is applied.

PullNet [11] is another solution that achieves great results, especially on our targeted data set **MetaQA**. In comparison to [10], this paper derives answers both from KB and from a corpus text. However, we will focus on its achievements for **QAKG** and its work on missing links. **PullNet** creates a *question-specific sub-graph*, by using an iterative process to learn how to "*pull*" facts and sentences from the available data. The pull operations can retrieve information from one of the knowledge sources proposed or extract entities from a fact or a document. As well as [10], it uses a **graph-CNN** for reasoning. The limitation of this model consists in the need of having an additional text corpus (often limited) to be able to answer a question when the KG is very sparse, because it takes longer paths to reach the answer. Moreover, it requires the answer to be an entity in the extracted *subgraph*. Both solutions achieved and surpassed the state of the art when they were promoted, [10] proving to be more efficient for the task and a solution we desire to explore further. An observation is that the *incomplete knowledge graph is created by removing randomly links from the original one*. A challenge would be to see how this randomness affects the results.

II. Question Answering

Question Answering is an intense research topic in the academia world ([10], [11], [19], [3], [2], [5], [16], [4]). There are a multitude of *machine learning and natural language processing techniques that aim to achieve state-of-the-art solutions for providing best in-class answers to input questions*. A large set of approaches are based only on natural language processing techniques, aiming to find the answer to the questions based on individual documents. In [2], it was presented a more advanced neural model that incorporates and reasons based on data scattered across documents and across multiple documents making the transition to an inference problem on a graph, where nodes are mentions to entities and edges signal relations such as within and *cross-document co-reference*. There are proofs that the model described in [2] learns to answer questions by collecting evidence from a variety of documents using a differential message passing algorithm that updates node representations based on their *proximity*.

[16] has proofed that **Question Answering** (QA) models over **Knowledge Bases** (KBs) are capable of providing more precise answers by utilizing relation information among entities. In the paper [16], the authors had proposed a *Relation-updated Direction-guided Response Selector* (**RDAS**) model, that learns structure information by converting links in each sub graph to additional nodes, which can be used as a path *knowledge* to improve the thinking abilities.

Some literature approaches has completely different approaches: the *multi-hop knowledge-based QA challenge*, [4] suggested a novel teacher-student solution. The student network in the methodology attempts to find the right response to the question, while the teacher network wants to learn intermediate supervision signals to improve the student network's thinking ability. Similar approach has been used in [5], where a generic neural state machine was proposed to focus on the task itself, while the teacher network aims to learn intermediate supervision signals to improve the student network, addressing an issue with *multi-hop* (the answer entities are multiple hops away from the topic entities in the knowledge graph) knowledge-based question-answering, which is that algorithms can only receive the feedback from the final answer. The main *drawback* of this learning process is that it becomes *extremely unstable*. **A Neural State Machine** [6] is a graph-based network that simulates the computation of a finite automaton, that follows 2 stages: modeling and inference, the first to construct the state machine, and the second to simulate its operation.

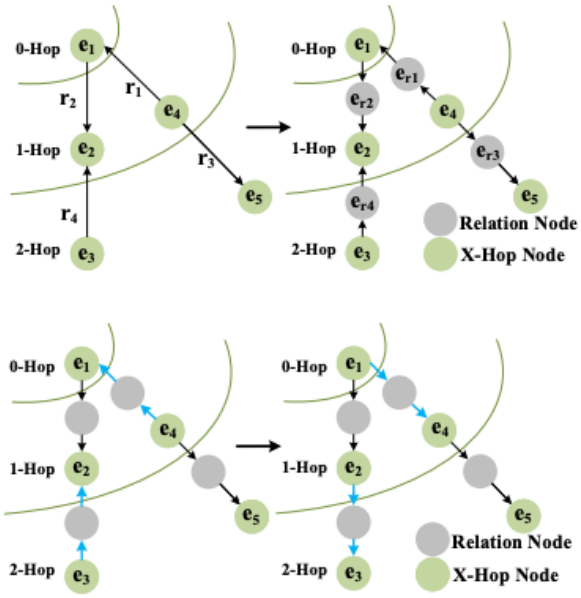

Figure 2: The integration process idea for direction information in [16]. This requires a prerequisite of transforming relation edges into relation nodes.

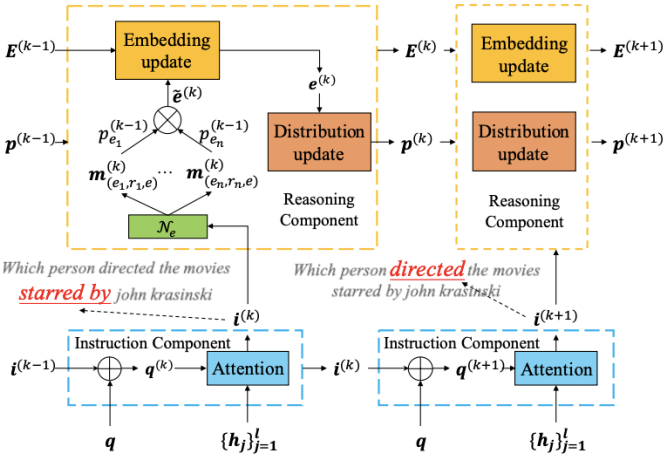

Figure 3: An overall sketch of NSM presented in [5]. This illustrates the two reasoning steps for NSM on the question *"which person directed the movies starred by john krasinski"*. In different reasoning steps, the instruction vector focuses on different parts of the question.

There is evidence in [5], using extensive experiments on three *benchmark* datasets that the effectiveness of this approach increased the overall results.

Zhang et al. [19] proposed an **end-to-end learning framework** for *question answering based on a knowledge graph* which is named variational reasoning network (**VRN**). In Figure 3, there can be observer the arhitecture consisting of *two modules*, one for **topic entity recognition** and one for **logic reasoning over knowledge graph**. The purpose of the probabilistic module for entity

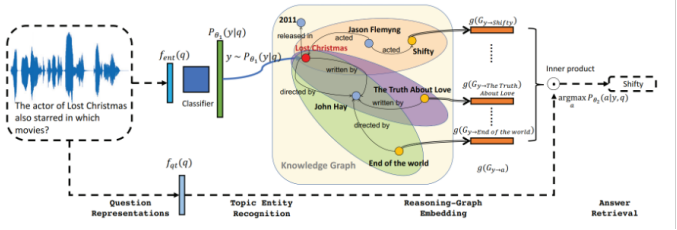

Figure 4: The architecture proposed for Variational Reasoning for Question Answering with Knowledge Graph in [19]

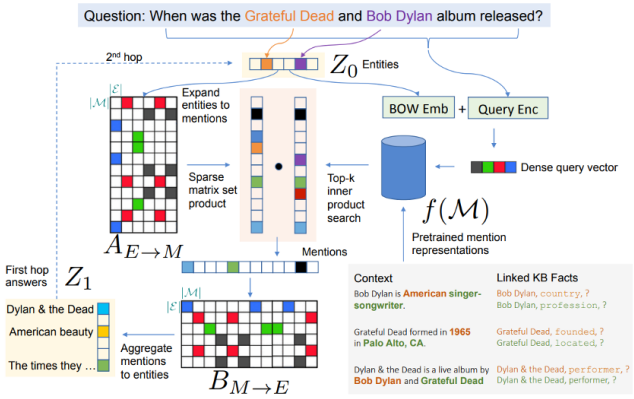

Figure 5: The solution for how DrKIT[3] module answers multi-hop questions by iteratively mapping an input set of entities to an output set of entities.

recognition is to handle the uncertain topics resulting from unlabeled topic entities. *Specifically*, it is used a neural network so that the question can be represented in a *d*-dimensional vector. The neural network can be either a recurrent neural network to embed sentences or a *simple embedding network mapping bag-of-words to a vector*. The module for logic reasoning over *knowledge graph* has a reasoning-graph embedding architecture, where all the inference rules are represented as **non-linear embeddings in vector space**.

The paper [3] describes a neural module named **DrKIT** which crosses through textual data represented like a knowledge base and, at each step, uses sparse-matric TF*IDF indices and a maximum inner product search on a special index of contextual representations of the mentions. The algorithm starts by creating a *set of words* from a specific question, and it aims to follow relevant outgoing relation edges in the knowledge base to arrive at the answer. The *set of words* is expanded to a set of co-occurring mentions and a neural network is trained to be capable of filtering the mentions based on a relevance score relative to the question. After *aggregating* the resulting set of mentions to the entities they refer to, it results an ordered set of entities which are **answer candidates results**.

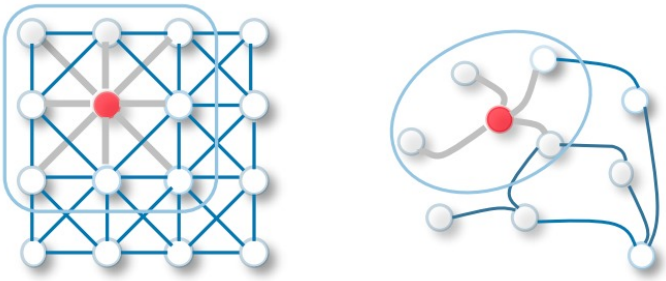

Figure 6: Representation of difference between CNN and GCN

III. Dataset Description

A. MetaQA

MetaQA[20] is a large dataset that contains around 400k single and multi-hope questions in the movie domain. The dataset is divided into three categories, however the one which is of interest for us is the "*vanilla*" version. The associated knowledge base consists of **135 triples**, **43k entities**, and **9 relations**. There are up to *21 types of multi-hope questions obtained from 10 templates*. The list of question type can be found in [20].

B. WebQuestionsSP

WebQuestionsSP[18] is smaller having around 5k questions (*1-hope* and *2-hope*) that are answerable through Freebase KB. This KB contains **24.9M** entities. The state of the art often restricts the KB to be a subset of **Freebase** which contains all facts that are within 2-hops of any entity mentioned in the questions of **WebQuestionsSP**.

C. Complex WebQuestions 1.1

Complex WebQuestions 1.1[13] is generated from **WebQuestionsSP** by extending the question entities or adding constraints to answers, in order to construct more complex **multi-hop questions**(4-hop on Freebase KB). It uses the same KB and corpus **WebQuestionsSP**.

IV. Technologies

A. Graph-CNN

Graph Convolutional Networks[7] perform similar operations as **CNNs**, where the model learns the features by inspecting neighboring nodes. The major difference between **CNNs** and **GNNs** is that **CNNs** are specially built to operate on regular (Euclidean) structured data, while **GNNs** are the generalized version of **CNNs** where the numbers of nodes connections vary and the nodes are unordered (*irregular on non-Euclidean structured data*).

B. Graph Embeddings

Knowledge Graph embeddings have been widely studied in recent years. Much like word embeddings find vector representations of words using text corpora, KG embeddings aim to find vector representations of entities and relations in a KG. The most popular application of KG embeddings is KG completion or finding true facts that are missing from the KG

1) ***ComplEx****[14]: is a tensor factorization approach that embeds relations and entities in complex space*. It is a solution linear both in space and time, and it is based on the Hermitian product, the standard real dot product counterpart in complex space. It treats each entity or relation embedding as a d-dimensional vector of complex numbers. The scoring function for triple (h,r,t) is the real part of the dot product of eh, er and the element wise conjugate of et.

2) ***TransR****[8]: is a representative translational distance model that represents entities and relations as vectors*, being able to project an entity to a different relationship semantic spaces. Its main disadvantage is adding the complexity of the model and having a higher rate of data transfer, which affects distributed training.

3) ***RotatE****[12]:* , being inspired by **TransE**, *is based on Euler's identity*. It defines relations as rotation from *head to tail*. Not only that is a *scalable* solution, but is also able to infer and model various relation patterns. **RotatE** performance is better than **ComplEx**, *the embedding method* used by [10].

C. Text Embeddings

RoBERTa [9] is a BERT based language model trained on over 160GB of text data. It is a variant of of BERT which "modifies key hyperparameters, removing the next-sentence pretraining objective and training with much larger mini-batches and learning rates".

We pass the question text into RoBERTa and use the hidden states of the last layer to get a 768-dim representation of the question.

V. Solution

We believe that the integration process idea for direction information in [16] is a promising solution, as it can translate an unstructured graph in a tree with unidirectional paths from roots to leafs. This is solved by transforming relation edges into relation nodes, and the n-Hop search is greatly simplified in the KG. We believe that this is worth trying for our approach. Another approach that we will try is to check the attention model proposed by [5] for updating the embeddings models. More embedding model worth exploring has been proposed in [17], [15].

In [10] and [11], there are proposed two methods for improving the question answering process over sparse knowledge graphs. **EmbedKGQA** [10] does not use an additional text corpus, but only the knowledge base and

Model	MetaQA KG-Full			MetaQA KG-50		
	1-hop	2-hop	3-hop	1-hop	2-hop	3-hop
VRN	**97.5**	89.9	62.5	-	-	-
GraftNet	97.0	94.8	77.7	64.0 (91.5)	52.6 (69.5)	59.2 (66.4)
PullNet	97.0	**99.9**	91.4	65.1 (92.4)	52.1 (90.4)	59.7 (85.2)
KV-Mem	96.2	82.7	48.9	63.6 (75.7)	41.8 (48.4)	37.6 (35.2)
EmbedKGQA (Ours)	**97.5**	98.8	**94.8**	**83.9**	**91.8**	**70.3**

Figure 7: Accuracy comparison of EmbedKGQA[10] with state of the art for MetaQA dataset

the questions in the dataset. From the results presented in the *state of the art*, we reached to the conclusion that **EmbedKGQA** is one of the solutions we want to analyze further. In the Figure 7, we can observe the comparison in performance for [10], [11], and [19].

For further research in the field of question answering using knowledge graphs, we asked ourselves some questions and for the last two milestone we are trying to answer to them. As a consequence we formulated the following objectives:

1) **O1: Create a baseline for our project**
   Starting from the idea proposed in [10], we are building our model. Firstly, we are going to train and test the model on MetaQA.
2) **O2: Determine if we can achieve the same performance as state of the art**
   For MetaQA, the current state of the art for the 2-hop reasoning is 99.9, obtained using PullNet. Taking this into consideration, is it only natural that we would want to investigate this.
3) **O3: Test the solution on other datasets.**
4) **O4: Draw conclusions**

A. *The problem*

Given a question, written in natural language, determine the answer to that specific question from a knowledge graph, composed from entities and relations between them. The questions are categorized into 1-hop, 2-hop and 3-hop. Single-hop questions requires a single fact, meaning that the answer is one edge away from the entity (direct or reverse).

Multi-Hop question answering requires more facts and a reasoning chain in this process. For now, we are not exploring on a bigger data set that has n-hop questions, with n being larger than 3.However, a problem appear on the graph is incomplete, the answer can be found, but not in 3 steps. A missing fact causes a breaking in the reasoning chain, therefore a correct answer will not be given. As we mentioned before, one of the solution proposed for the multi-hope QA challenges were to use an additional corpus text, which may not be always available.

B. *Project Architecture*

Regarding the project architecture, our final scope is to be able to give one question as an input and be able to receive an answer for it. As a consequence, we have three main components:

1) **The model generator**
   Here we implement the model, we train it and we evaluate.
2) **The data generator**
   It encodes both the NLP questions and the knowledge graph. For the graph embeddings we are using ComplEx, this choice is going to be explained in the next chapter of this paper.We did not train any embeddings, we used pre-trained models. For the text embeddings, we are using a Pytorch pre-trained RoBERTa and for the graph we used the public embeddings for entities and relations trained by Saxena et al [10].
3) **The Engine**
   The engine takes as an input, it pre-processes this question and feeds it to the pre-trained model in order to receive an answer.
4) **The Flask Server**
   A small server that allows us to connect the Machine Learning module with the graphic interface.
5) **The frontend**
   We've implemented a small UI that contains a search bar that accepts as an input a NLP question and sends it to the Flask server that forwards it to the engine to obtain the answer. The answer travels back to the frontend application, where a call to the Google Knowledge Graph is done, so we can display more information about that specific question. This, actually, helps us to verify if an answer is actually correct, since the dataset is not updated, (MetaQA) The frontend component is written in React using CSS elements.

C. *The model*

For the model architecture we try to reproduce the implementation described in [10], This represent our starting point for improving and analyzing further.

Our model is in fact a LSTM that receives as input the question embbeded using RoBERTa. The question will be further projected in the complex space, in order to have the same dimension as the entities.After getting the head embedding (from KG embedding module) and question, a score is calculated for all candidate answers (which is all entities in the KG). The scoring function it is the ComplEx scoring function. Highest scoring entity is chosen as the answer. The entity embeddings are used for learning a triple scoring function between the head entity, question, and answer entity.

A scoring function which ranks each relation for a given question q is learn by our model.The scoring function is

Model	1-hop	2-hop	3-hop
KRAM-roBERTa	21.3%	31.2%	15.6%
KRAM+roBERTa	97.8%	98.2%	94.5%

Figure 8: Results for KRAM withous RoBERTa and KRAM with RoBERTa

defined as the sigmoid of the dot product of the final output of the last hidden layer of RoBERTa and the embedding of relation r.

VI. Results

We've used for training mainly Adam optimizer with a learning rate equal to 0.0005, stopping the model if no improvement observed in 5 epochs. As a loss function we used Binary Cross Entropy and we trained on a NVIDIA GTX 1060 8GB GPU.

For the third milestone of this project we trained the model on the Vanilla text data from the MetaQA. This part of the dataset is composed from the following components:

1) 1-hop: derived from the wiki-entities branch of the Facebook MovieQA
2) 2-hop: 21 question types with 10 text templates for each type
3) 3-hop: 15 question types with 10 text templates for each type (2-hop and 3-hop are generated from the movie knowledge base in MovieQA)

The first attempt of training was with a model that did not use embeddings on the questions and the accuracy on the test set was very low. What is iteresting is that the training accuracy was still good, around 90 percent. For the training we used the Adam optimizer with a learning rate equal to 0.0005. The training accuracy was high, around 99 percent for every subset. We trained the model for 1-hop, 2-hop and 3-hop using a model that is embedding the question using RoBERTa. The results can be see in Figure 7 and 8.

The next step was to validate our model performance on WebQuestionsSP. In this case we choose to stay true to the dataset used in the state of the art. The KB is restricted to be a subset of Freebase which contains all facts that are within 2-hops of any entity mentioned in the questions of WebQuestionsSP. Furthermore, only those relationship that are present in the dataset are ketp. This smaller KB has 1.8 million entities and 5.7 million triples.

Due to the lack of proper computational power and the not so conclusive results for this dataset we did not proceed further with the investigation because it was clear to us that the algorithm needs to be revised. This can be seen in Figure 10. .We concluded that it cannot achieve great results for more than 3-hope reasoning and for more complex questions than the templates defined in the previous dataset used (MetaQA).

KGE	KRAM + roBERTa
ComplEX	98.2%
RotatE	97.8%
TuckER	97.3%

Figure 9: Results with differents knowledge graph embeddings

Model	1-hop MetaQA	2-hop MetaQA	3-hop MetaQA	2-hop MetaQA-half	2-hop WebQuestionsSP
PullNet	97.0%	99.9%	91.4%	52.1%	46.7%
KV-Mem	96.2%	82.7%	48.9%	41.8%	68.1%
EmbedKGQA	97.5%	98.8%	94.8%	91.8%	66.6%
KRAM+roBERTa	97.8%	98.2%	94.5%	84.7%	39.5%

Figure 10: Results with differents knowledge graph embeddings

We reach the conclusion that the MetaQA has a small number of entities compared to other datasets and this can be a reason for which the accuracy has a high value. Moreover, the types of questions in the train and test sets are in similar order.

VII. Conclusions and future work

The main conclusion we draw from this project is that most of the models, including the state of the art ones work well on specific dataset or a specific knowledge base. Not using an additional text corpus, our algorithm is not able to learn the answer for new questions, it needs to know the type of question and the words used. Moreover, if the head entity is not found in the vocabulary, the algorithm will fail by generating an exception. We simplified the task by giving a direction to the graph, starting with a head entity. Further research are needed for the graphs that don't contain all the entities from the training/test dataset.

This task is particularly demanding with the computational resources, without a powerful GPU is hard to conclude the model performances, due to the lack of variety in training. Overall, this project represented for us a great learning opportunity and gave us the chance to analyze better this domain of research. For sure it is hot topic for the ACL papes. It will be interesting to see how the generalization task progress over the time or if the research is still dedicated to a specific dataset.

As a general conclusion, by following the indication from [10], we were able to produce a close to the state of the art model named KRAM, which has fairly good performance on the MetaQA dataset.

References

[1] Introducing the knowledge graph: things, not strings. https://blog.google/products/

search/introducing-knowledge-graph-things-not/. Accessed: 2021-04-11.

[2] Nicola De Cao, Wilker Aziz, and Ivan Titov. Question answering by reasoning across documents with graph convolutional networks. *arXiv preprint arXiv:1808.09920*, 2018.

[3] Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W Cohen. Differentiable reasoning over a virtual knowledge base. *arXiv preprint arXiv:2002.10640*, 2020.

[4] Tiantian Gao, Paul Fodor, and Michael Kifer. Querying knowledge via multi-hop english questions. *arXiv preprint arXiv:1907.08176*, 2019.

[5] Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. *arXiv preprint arXiv:2101.03737*, 2021.

[6] Drew A Hudson and Christopher D Manning. Learning by abstraction: The neural state machine. *arXiv preprint arXiv:1907.03950*, 2019.

[7] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[8] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[9] 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. *arXiv preprint arXiv:1907.11692*, 2019.

[10] 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, 2020.

[11] Haitian Sun, Tania Bedrax-Weiss, and William W Cohen. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537*, 2019.

[12] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

[13] Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*, 2018.

[14] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080. PMLR, 2016.

[15] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[16] Xu Wang, Shuai Zhao, Bo Cheng, Jiale Han, Yingting Li, Hao Yang, Ivan Sekulic, and Guoshun Nan. Integrating subgraph-aware relation and direction-reasoning for question answering. *arXiv preprint arXiv:2104.00218*, 2021.

[17] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[18] Wen-tau Yih, Matthew Richardson, Christopher 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, 2016.

[19] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander Smola, and Le Song. Variational reasoning for question answering with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

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