Relational Graph Reasoning for Knowledge-Augmented Question Answering
Final Report

Jun Yan
University of Southern California
yanjun@usc.edu

Abstract

Pretrained language models have been widely used in various Natural Language Processing (NLP) tasks and achieved remarkable success. However, there are two shortcomings of such methods: (1) while language models do well in encoding word sequence based on its semantic meanings, it can't introduce knowledge from other sources, which limits its performance on knowledge-guided NLP tasks; (2) language models understand semantics based on co-occurrence in the training corpus, which makes it hard to do complex reasoning.

In this paper, we focus on the question answering task where external knowledge is necessary for both understanding the context and identifying the correct answer. Inspired by Relation Network (Santoro et al., 2017), we propose a framework to incorporate relevant facts from knowledge graph and do reasoning. Experiments on CommonsenseQA dataset demonstrate the effectiveness of our method and the value of external knowledge.

1 Introduction

When humans use their languages to communicate with each other, they often rely on broad implicit assumptions, such as factual knowledge and commonsense knowledge. Humans learn and use this kind of assumptions in everyday life, which make their language concise without lacking precision. However, machines by nature don't have such background knowledge. Machine learning models can't accumulate human's commonsense knowledge or domain-specific factual knowledge through interacting with the environment. Therefore, empowering Natural Language Processing (NLP) techniques with knowledge is one of the major long-term goals for Artificial Intelligence (AI).

Question Answering (QA) is a Natural Language Understanding (NLU) task requiring both language processing and knowledge reasoning. While many works focus on how to efficiently find support evidence from the context, we argue that external knowledge is also important for answering questions. In this work, we propose a knowledge-augmented question answering framework that takes an external knowledge graph as resources. For each question-answer pair, we first collect helpful evidence from the knowledge graph. The knowledge is then organized as an evidence graph. After that we learn a network that can do relational reasoning on the evidence graph to answer the question.

Specifically, in this work, we choose commonsense question answering as our task. The main focus of the commonsense question answering task is to incorporate commonsense knowledge and conduct reasoning. We choose CommonsenseQA (Talmor et al., 2018) as the dataset, and use ConceptNet (Speer et al., 2017) as the knowledge resources. Details will be given in the next section.

2 Background

In this section, we will introduce the dataset, knowledge source, and formally present our task.

2.1 CommonsenseQA

We use CommonsenseQA (Talmor et al., 2018) as our training and testing datasets. It contains 12k multiple choice questions asking for a target concept from ConceptNet. The statistics are showed in Table 1.

	train	dev	test
# instances	9,741	1,221	1,140

Table 1: Statistics of CommonsenseQA dataset (official split).

Note that the ground truth label for the official split test set is not provided. Therefore, to evaluate our model, we instead use in-house split as in Lin et al. (2019). Specifically, we split the original train set into in-house train set and in-house test set. The statistics are showed in Table 2.

	train	dev	test
# instances	8,500	1,221	1,241

Table 2: Statistics of CommonsenseQA dataset (in-house split).

2.2 ConceptNet

ConceptNet (Speer et al., 2017) is a multilingual knowledge graph whose nodes are concepts in the form of words or phrases and edges are relations between connected nodes. We use the English part of ConceptNet 5.60 as our knowledge resources. When preprocessing the graph data, we:

(1) merge the original 42 relation types into 17 types based on their semantic meanings to relieve sparsity issue. (e.g. *RelatedTo*, *SimilarTo*, *Synonym* can be merged.)

(2) introduce reverse relations to expand existing relations (e.g. *AtLocation*$^{-1}$ with *AtLocation*).

The statistics of the preprocessed knowledge graph is showed in Table 3.

# nodes (concepts)	# edges (facts)	# type of edges (relations)
799,273	2,487,810	34

Table 3: Statistics of preprocessed ConceptNet knowledge graph.

2.3 Problem Setup

In the problem of commonsense question answering, given a question q, the model is asked to select the correct answer a_k from a set of candidate answers $\{a_i\}$. Here is an example:

> *A revolving door is convenient for two direction travel, but it also serves as a security measure at a what?*
>
> A.bank B.library C.department store
> D.mall E.new york

Meanwhile, we also have a knowledge graph $G = (V, E)$ serving as the knowledge source, where V is the node set and E is the edge set. Each node is a concept and each directed edge indicates the relation between two nodes that it connects with.

3 Framework Overview

Figure 1 gives the overview of our proposed framework. We convert the question answering problem into measuring the plausibility between the question and each candidate answer. The answer with the highest plausibility score will finally be chosen. Therefore, the input of our framework is a question-answer pair, and the output is a plausibility score between 0 and 1.

Given a question-answer pair, our workflow can be divided into 4 stages: concept grounding, graph building, triple encoding and attentive pooling.

3.1 Concept Grounding

Given question q and answer a, we use SpaCy[1] to perform pattern matching to link each concept in the word sequence of q to concept $c^q_1, \ldots, c^q_l$, and we map a to concept $c^a_1, \ldots, c^a_m$, where l is the number of concepts mentioned in q and m is the number of concepts mentioned in a. Note that word overlapping is allowed when doing matching. For example, from "apple tree" we can extract three concepts: "apple tree", "apple" and "tree". We then have $l + m$ concepts $c^q_1, \ldots, c^q_l, c^a_1, \ldots, c^a_m$ in the question-answer pair that have been grounded to the knowledge graph.

3.2 Graph Building

After we identify the concepts for a question-answer pair, we want to retrieve knowledge from knowledge graph G to help understanding the context. Formally, we want to extract a subgraph from G with grounded concepts $c^q_1, \ldots, c^q_l, c^a_1, \ldots, c^a_m$ as nodes. We name it as *Evidence Graph* since it's a subgraph about evidence contextualized to a given question-answer pair.

After we locate the grounded concepts in the knowledge graph G, we find **short paths** between any question concept and answer concept to add nodes and edges into the evidence graph. We assume that two concepts have informative relations only when they can be connected within k-hop relations, where k is a hyperparameter. In other words, we find paths between any question concept and answer concept with length up to k and add them to our evidence graph.

[1] https://spacy.io/

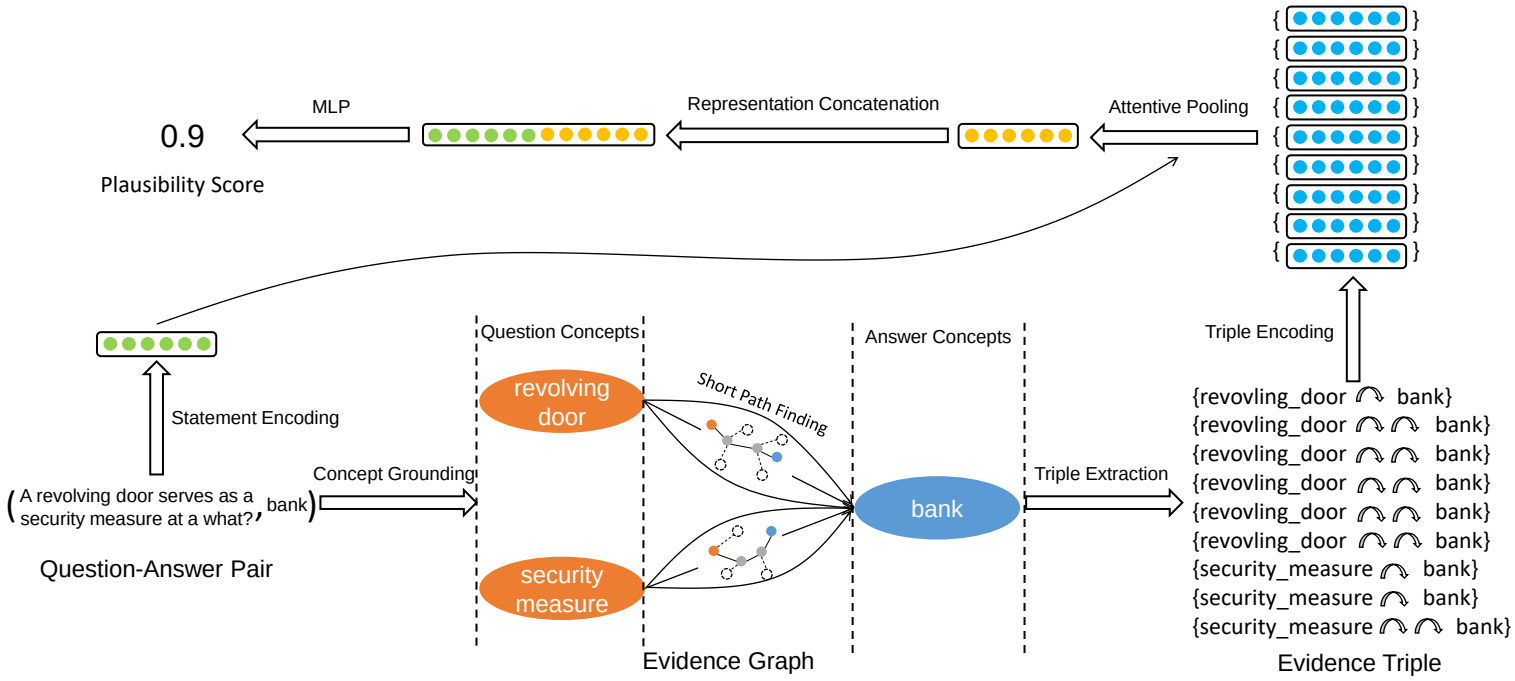

Figure 1: Overview of Our Framework.

For example, in the question mentioned in Section 2.3, we ground two question concepts "revolving door", "security measure" and one answer concept "bank". Some short paths between two question-answer concept pairs are:

- $\textbf{revolving\_door} \xrightarrow{AtLocation} \textbf{bank}$
- $\textbf{security\_measure} \xrightarrow{Synonym} \textbf{security\_system} \xrightarrow{Synonym^{-1}} \textbf{security} \xrightarrow{RelatedTo^{-1}} \textbf{bank}$

3.3 Triple Encoding

Inspired by the idea of Relation Network (Santoro et al., 2017), we regard paths between question and answer concept as basic units in the evidence graph and assume that the evidence between question and answer concept is contained on the edges of the path. That means, given a question concept and an answer concept, we only consider multihop relation between them and neglect the intermediate nodes. Then, the evidence paths can be converted into evidence triples where the head and tail concepts remain and the new relation is either a basic relation (when the path indicates one-hop relation) or a compositional relation (when the path indicates multi-hop relation).

For example, one-hop evidence path

$$\textbf{revolving\_door} \xrightarrow{AtLocation} \textbf{bank}$$

will be converted into evidence triple

$$(\textbf{revolving\_door}, \mathit{AtLocation}, \textbf{bank}).$$

Three-hop evidence path

$$\textbf{security\_measure} \xrightarrow{Synonym} \textbf{security\_system} \xrightarrow{Synonym^{-1}} \textbf{security} \xrightarrow{RelatedTo^{-1}} \textbf{bank}$$

will be converted into evidence triple

$$(\textbf{security\_measure}, \mathit{Synonym} \circ \mathit{Synonym}^{-1} \circ \mathit{RelatedTo}^{-1}, \textbf{bank}),$$

where $\circ$ denotes relational composition.

After we extract evidence triples from the evidence graph, we want to encode them into triple embeddings to support further operation. The triple embedding is obtained by concatenating head concept embedding, relation embedding, and tail concept embedding followed by a linear transformation. Head concept embedding and tail concept embedding shares the same embedding lookup matrix. Relation embedding is also obtained by looking up in an embedding matrix. Denote the number of basic (one-hop) relations as n_r, then when we take into consideration up

to k-hop compositional relations, there will be $n_r + n_r^2 + \ldots + n_r^k$ possible relations.

Formally, for an evidence triple (c_q, r, c_a), the triple encoding is calculated as:

$$\mathbf{t} = \text{Linear}([\mathbf{c_q}; \mathbf{r}; \mathbf{c_a}]),$$

where $\mathbf{c_q}$, $\mathbf{r}$, and $\mathbf{c_a}$ are embeddings of concept c_q, relation r, and concept c_a respectively. $[\cdot;\cdot;\cdot]$ means vector concatenation. Linear denotes a linear transformation with trainable weights.

3.4 Attentive Pooling

The triple encoding stage has converted evidence from the knowledge graph to a set of triple embeddings, each one corresponding to a reasoning path between a question concept and an answer concept. We then want to aggregate those triple embeddings to get the representation for the graph-based evidence. Specifically, suppose we have n evidence triples and their embeddings are $\mathbf{t_1}, \mathbf{t_2}, \ldots, \mathbf{t_n}$. Our motivation is to select paths that are crucial for question answering while alleviating the noise, therefore we want to do attentive pooling over $\mathbf{t_1}, \mathbf{t_2}, \ldots, \mathbf{t_n}$ to get a representation vector.

To help identify important paths, we need a global signal from the question-answer pair. To better capture its semantic meanings, we adopt pretrained language models (e.g. BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019b)) to encode the question-answer pair:

$$\mathbf{h_{qa}} = \text{LM}([CLS], q, [SEP], a, [SEP])$$

where LM denotes a pretrained language model, $[CLS]$ and $[SEP]$ are special tokens as introduced in Devlin et al. (2018). $[\cdot,\cdot]$ means token concatenation.

Then we adopt multi-head dot-product attention mechanism (Vaswani et al., 2017) to select from $\mathbf{t_1}, \mathbf{t_2}, \ldots, \mathbf{t_n}$ attending to $\mathbf{h_{qa}}$ and get the graph the graph representation $\mathbf{h_{graph}}$:

$$\mathbf{h_{graph}} = \text{MultiHead}(\{\mathbf{t_1}, \mathbf{t_2}, \ldots, \mathbf{t_n}\}, \mathbf{h_{qa}}).$$

After that, we concatenate $\mathbf{h_{qa}}$ and $\mathbf{h_{graph}}$, combining information from both the question-answer pair and the external knowledge graph, and evaluate the plausible score by feeding the concatenated vector into an MLP (multilayer perceptron) followed by an activation function.

$$s_{qa} = \sigma(\text{MLP}([\mathbf{h_{qa}}; \mathbf{h_{graph}}])).$$

We can then do supervised learning with the predicted plausible score s_{qa} and the gold label l_{qa} using binary cross-entropy loss loss, where $l_{qa} = 1$ indicating that a is the correct answer to q and $l_{qa} = 0$ otherwise:

$$loss = -l_{qa} \log(s_{qa}) - (1 - l_{qa}) \log(1 - s_{qa}).$$

4 Experiments

4.1 Experiment Setup

Pretrained Embedding. We pretrain Trans-E (Bordes et al., 2013) embeddings on ConceptNet to get pretrained concept embedding and basic relation embedding. For higher-order relation, we set their embedding as the summation of basic relation embeddings. All concept embeddings and relation embeddings are finetuned during training.

Hop Threshold k. Threshold k controls how many paths do we want to incorporate in reasoning. It also reflects our judgement on the relationship between path length and noise. We test $k = 2$ and $k = 3$ in our experiments. Note that we need to set a hop threshold to collect relational paths instead of always finding the shortest paths. Because finding shortest paths calls for an algorithm to run on the global graph structure in time $O(|E| + |V| \log |V|)$ (Dijkstra's algorithm) where both $|E|$ and $|V|$ are huge numbers for a knowledge graph. On the contrary, when we set the hop threshold k, we can only rely on local structure (extend to k-hop neighbors of a given node), which makes path finding computational feasible.

Statement Encoding. We test our proposed framework with three representative pretrained language models: BERT-BASE, BERT-LARGE (Devlin et al., 2018), and ROBERTA-LARGE (Liu et al., 2019b). We build these text encoder based on Huggingface's transformers library.[2]

Optimization. We use RAdam optimizer (Liu et al., 2019a). For text encoders, we set the learning rate to 3×10^{-5} for BERT-BASE, 2×10^{-5} for BERT-LARGE, and 1×10^{-5} for ROBERTA-LARGE. For graph encoders, we set the learning rate to 3×10^{-4} for our proposed method and 1×10^{-3} for RGCN.

4.2 Compared Methods

LM Finetuning. Pretrained language models can be directly adopted to the question answering task

[2] https://github.com/huggingface/transformers

in a knowledge-agnostic way. Specially, we fine-tune a binary classifier on top of the pretrained language model encoder to calculate the plausible score.

Relational Graph Convolutional Network (RGCN). Traditional Graph Convolutional Network (GCN) (Kipf and Welling, 2016) leverage both node features and local graph structures to do message passing and learn hidden representation of the graph. Schlichtkrull et al. (2018) propose relational graph convolutional networks (R-GCNs) to model relational graph data. It assigns different weight matrices to edges of different relational types and can only deal with a closed set of relations. RGCN can be applied to encode the evidence graph, thus serving as the baseline of the graph encoding part of our method.

Knowledge-Aware Graph Network (KagNet). KagNet (Lin et al., 2019) can be regarded as an extension of RGCN. It uses LSTM to encode evidence paths and adopts both concept pair level attention and path level attention. It achieved state-of-the-art results on CommonsenQA dataset when it's submitted.

4.3 Experimental Results

We compare our model with baseline models as described in Section 4.2 on CommonsenseQA dataset. We use the in-house split as in Lin et al. (2019). We run all models only once except that we directly use the results reported in Lin et al. (2019) for KagNet. The results are presented in Table 4. We can observe that:

- Our model ($k = 2$) consistently outperforms other models when using BERT-Base and BERT-Large as the text encoder, which demonstrates the effectiveness of our model.
- The improvement brought by using external knowledge becomes marginal when we switch to more powerful pretrained language models. Our hypothesis is that when we have more training corpus, current language models have the capacity of memorizing the information contained in the corpus. In this case, graph knowledge is no longer needed. Therefore, incorporating graph knowledge doesn't necessarily show improvement.
- With BERT-BASE and BERT-LARGE, for our model, $k = 2$ consistently performs better that $k = 3$. That may be due to paths with lengths longer than 2 are mostly noisy, which means these paths indicate only weak relations between concepts of interests. Therefore, incorporating them in the reasoning process hinder the attention mechanism from selecting the most valuable reasoning paths and leads to performance drop.
- When using RoBERTa-Large as the text encoder, the performance of our model with $k = 2$ significantly drops compared to other models. That may be due to randomness in the training process that makes training stop too early.

4.4 Study: High-Order Relation Embedding

Since we use a heuristic way to initialize the embedding for high-order relation embedding, we also want to see if the underlying assumption makes sense for compositional relations. That is, should the embedding of high-order relation be always the same as the summation of the corresponding basic relation embeddings. To test this, we introduce another design "-Tying", which means tying the embedding of high-order relation to be the sum of basic relation embedding throughout the training process instead of just for initialization.

Results are showed in Table 5. It can be seen that adding the tying constraint leads to performance degradation. That implies that multi-hop relations cannot be simply modeled as the composition of basic relations. They're likely to carry on extra information. As the training goes, the TransE assumption may not hold and we should assign an independent embedding for each high-order relation.

5 Related Work

Knowledge Extraction. There are two main sources to extract related knowledge from: plain text and knowledge graph. Chen et al. (2017) propose DrQA which consists of a document retriever and a document reader to locate and incorporate helpful knowledge in the Wikipedia. Many knowledge-augmented works (Bauer et al., 2018; Lin et al., 2019) directly use a knowledge graph in the related domain. Lv et al. (2019) extract helpful knowledge from both ConceptNet and Wikipedia. While text data like Wikipeida has high coverage, structured data like knowledge graph can provide

Methods	BERT-Base		BERT-Large		RoBERTa-Large	
	Dev (%)	Test (%)	Dev (%)	Test (%)	Dev (%)	Test (%)
LM Finetuning	48.81	46.49	62.41	57.21	73.32	**69.62**
RGCN	56.84	54.71	62.74	56.41	72.40	68.09
KagNet†	55.57	56.19	62.35	57.16	-	-
Ours ($k = 2$)	**58.80**	**57.61**	**64.21**	**60.11**	24.90	23.85
Ours ($k = 3$)	56.84	53.99	62.00	55.84	**73.79**	68.73

Table 4: **Performance comparison of our models and baseline models on CommonsenseQA dataset.** We report accuracy on the in-house dev set and in-house test set as in Lin et al. (2019) on CommonsenseQA. Each model is run once. † indicates reported results.

Methods	BERT-Base		BERT-Large	
	Dev (%)	Test (%)	Dev (%)	Test (%)
Ours ($k = 2$)	**58.80**	**57.61**	**64.21**	**60.11**
Ours-Tying ($k = 2$)	58.97	57.05	63.88	58.90

Table 5: **Performance comparison of whether to use tying for high-order relation embeddings.**

relation information which is necessary for knowledge reasoning. In this paper, we focus on leveraging structured knowledge.

Given the knowledge source and input, the next step is to extract related knowledge. This task is called concept grounding or entity linking. Knowledge-augmented QA papers usually use off-the-shelf tools (e.g. entity linker) or develop simple string matching rules to identify matched entities/concepts on the knowledge graph. After locating these "root" nodes, an extractive way to get the evidence graph is to construct a subgraph covering all "root" nodes. However, finding the minimal spanning subgraph is a NP-complete problem. Therefore, researchers develop heuristics (Lv et al., 2019; Lin et al., 2019) for graph construction or formulate the path finding problem as an optimization problem which can be efficiently solved. In this paper, we adopt simple path finding strategy that only finds paths within k-hops.

Graph Reasoning. After we have an evidence graph, the next step is to reason over it. Many Graph Neural Network (GNN) variants (Wu et al., 2019; Zhou et al., 2018) can be adopted for reasoning. Marcheggiani and Titov (2017); Zhang et al. (2018) find that Relational-GCN (Schlichtkrull et al., 2018) tends to over-parameterize the model. (Lv et al., 2019; Lin et al., 2019) both use GCNs on the undirected graph while Lin et al. (2019) propose an additional LSTM-based path encoder. The lesson here is that we should not only use symbolic knowledge from the graph, but also leverage semantic clues from the input sequence. Xiao et al. (2019) propose Dynamically Fused Graph Network to deal with constructed graph. It shows that entity-level reasoning and token-level contexts are both important for question answering. For these graph neural networks, the attention mechanism is usually helpful for feature aggregation as well as interpreting the results and debugging. In this paper, we develop our framework from Relation Network (Santoro et al., 2017), which is more efficient when tackling small relational graphs since message passing is not needed.

6 Conclusion

In this paper, we propose a novel framework for incorporating relevant knowledge from external knowledge graph to help knowledge-guided question answering. The workflow can be divided into four stages: concept grounding, graph building, triple encoding, and attentive pooling. Our framework consistently outperform other baseline models when using BERT as the text encoder.

References

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. *arXiv preprint arXiv:1809.06309*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *arXiv preprint arXiv:1909.02151*.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2019a. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. 2019. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering. *arXiv preprint arXiv:1909.05311*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826*.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*.

Yunxuan Xiao, Yanru Qu, Lin Qiu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. *arXiv preprint arXiv:1905.06933*.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. *arXiv preprint arXiv:1809.10185*.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*.