Reasoning Paths Generation for Commonsense Question Answering

Peifeng Wang

Abstract

Commonsense question answering (QA) requires a model to acquire some necessary background knowledge about how the world operates and people interact with each others. A large number of works have been dedicated to resort the help from commonsense knowledge graphs. However, these methods do not consider the context of the questions while retrieving evidence from knowledge graphs. Moreover, knowledge graphs are known to be far from complete and might not contain the necessary knowledge for answering the questions. In this paper, we propose to learn a reasoning paths generator to generate structured evidence dynamically according to the questions. Our generator uses pre-trained language model as the backbone, leveraging the tremendous unstructured knowledge stored in the language model to alleviate the incompleteness of knowledge graph. In addition, we propose to further train our whole framework to solve downstream tasks in a end-to-end fashion, greatly adjusting our generator to better serve the QA system. We further conduct experiments on commonsense QA dataset to demonstrate the effectiveness of our method.

1 Introduction

Most of the traditional question answering (QA) tasks are fact-based, which means that the QA systems could solely rely on some factoid evidence to answer the questions. For example, the machine reading comprehension based QA datasets usually provide some context as the only necessary source of evidence for answering the questions. They aim to test a system's ability of answers extraction over a given context (Rajpurkar et al., 2016). For open domain QA datasets, they require the systems to retrieve relevant evidence via some Information Retrieval (IR) methods from large corpus like Wikipedia. In this case, besides answers extraction, they also test a system's ability for fact finding at scale (Chen et al., 2017).

On the contrary, commonsense QA datasets including Commonsense QA (Talmor et al., 2018), Social IQA (Sap et al., 2019b) and CosmosQA (Huang et al., 2019), require reasoning over not only factoid evidence but also some background knowledge. Such background knowledge often refers to commonsense about how this world operates or how human interacts with each other. For example, to answer the question "Where would you find fungus growing on something made from milk?", we need commonsense knowledge like cheese is made from milk and fungus often grows on outdated food. Such knowledge is obvious for humans but not easy for most of the QA systems to retrieve (Talmor et al., 2018).

One direct solution for addressing these commonsense QA datasets is to leverage the knowledge graph (KG) for commonsense like Conceptnet (Speer et al., 2017) and ATOMIC (Sap et al., 2019a). Such KGs provide valuable facts on commonsense to complement the current QA systems. However, leveraging commonsense KG is nontrivial and poses the following challenges.

1. **Enormity**. A knowledge graph often consists of millions of triplet facts and only a few of them are relevant to the target questions.

2. **Sparcity**. Knowledge graphs are also known to be extremely incomplete. They might lack the facts which are necessary for answering the questions.

Previous works (Lin et al., 2019; Lv et al., 2019) on addressing the first challenges usually seek the help from some entity linking systems and graph searching algorithms. They firstly recognize the mentioned entities and link them to the KG. Then they construct a subgraph by searching the paths between these entities. The quality of the constructed subgraph relies heavily on the performance of the entity linking systems and sampling strategies, which brings the risk of error-propagation. For the second challenge, many efforts have been put for commonsense KG completion (Li et al., 2016; Bosselut et al., 2019). Nonetheless, they are not question-oriented, meaning that the newly-added knowledge facts still fail to cover the ones necessary for answering the questions. A recent work (Fu et al., 2019) augments the KG on the fly but still faces the first challenge.

To address both of the challenges, we propose a context-conditioned reasoning paths generation framework for solving commonsense QA. The proposed framework learns to generate reasoning paths dynamically based on the given question and answer choices for further reasoning, which aims at solving the first challenge. As for the second challenge, our framework adopts the pretrained language model as the backbone. This allows us to take advantage of the tremendous unstructured knowledge which is encoded in the large language model to complement the incomplete KG. Later, the framework employ a simple classifier to fuse both structured and unstructured evidence to make the final decision.

We summarize our contributions as follows.

1. We propose to fine-tune generative language model on sampled paths from KG to obtain a reasoning path generator. The paths generator serves for building structured knowledge dynamically as external evidence for solving commonsense QA.

2. We propose a end-to-end training framework for our model, which efficiently fuses information from both structured and unstructured knowledge for reasoning.

3. We perform an empirical study on the performance of our model in different setting. Currently, our performance over fine-tuned language models are not significant, indicating the room for improvement of our whole framework.

2 Background and Problem Setting

Unlike traditional KGs which have a well-defined space of entities and relations, KGs for commonsense like Conceptnet focus on modeling entities as natural language phrases and relations as open domain concepts (Bosselut et al., 2019). This also leads to the difficulty for grounding the subgraph from a KG. Still, an observed commonsense KG could be framed as $\mathcal{G} = (\mathcal{E}, \mathcal{R})$. Commonsense facts are in the form of triplets (s, r, o), where $s, o \in \mathcal{E}$ represent the entity phrases while $r \in \mathcal{R}$ represents the relation between them.

We consider such a problem setting where given a question (q), the task needs the system to select one of choices $(\{a_i\})$ as the right answer. The commonsense QA datasets mentioned above all fall into this setting. More importantly, these QA datasets require commonsense knowledge where commonsense KG might offer some but not all of the help.

In this paper, we focus on training a graph generator which generates a KG subgraph dynamically for each of the question such that the subgraph (1) is informative for answering the question and (2) contains knowledge facts which is missing from the sparse KG. In particular, we reduce the subgraph generation problem into multi-path generation problem. Assume that we employ a reasonable entity linking system which extract all the entities mentioned in the questions $\{e_i^q\}$ and those mentioned in the answer choices $\{e_j^a\}$. Our path generator **G** learns to output a reasoning path which connects one question entity and one choice entity. We argue that the message passing in a subgraph represented by these paths also facilitates the reasoning process. Moreover, this reduction allows us to leverage the power of the recent pre-trained language models, and we use GPT-2 (Radford et al., 2019) in this paper.

3 Subgraph Generation

The workflow of training a path generator **G** for solving commonsense QA consists a pre-training [1] stage and fine-tuning stage as follows.

3.1 Learning to Connect Entities

The goal of the pre-training stage is to teach our graph generator to generate a reasoning path which connects e_i^q to e_j^a. This path is supposed to encode a snippet of a commonsense KG which we rely on to boost the QA system.

We firstly sample a bunch of triplet paths via random walk on the commonsense KG as our pre-

[1]The pre-training stage here is different from the one in language models.

Table 1: Some sampled paths from Conceptnet.

(fucosidases, formof, fucosidase, derivedfrom, fucoside, derivedfrom, fucose, _formof, fucoses)
(spot_candy_aisle, _hassubevent, buying_fresh_fruits_and_vegetables, hassubevent, meet_other_people, _hasprerequisite, socialize, _relatedto, play_street)
(cyanolipids, formof, cyanolipid, relatedto, nitrile, _relatedto, octakis, _derivedfrom, octakishexahedron)
(new_invention_to_made,_causes,creating_idea,hassubevent,scratch_head,_hassubevent,cogitating,hassubevent,scratching_head)
(chess_pieces_fall_over,_causes,playing_chess,hassubevent,think,_relatedto,vessel,_relatedto,patrol_boat)

Algorithm 1 Paths Sampling

```
Input: $\mathcal{G} = (\mathcal{E}, \mathcal{R})$
Output: A set of triplet paths $\{p\}$.
 1: repeat
 2:   current_node $u \leftarrow uniform\_sample(\mathcal{E})$
 3:   $p \leftarrow \{u\}$
 4:   for $t = 1$ to $T$ do
 5:     $N \leftarrow Neighbor(u)$
 6:     next_node $v \leftarrow uniform\_sample(N)$
 7:     $M \leftarrow All\_Relations(u, v)$
 8:     $r \leftarrow uniform\_sample(M)$
 9:     $p \leftarrow p \cup \{r, v\}$
10:     $u \leftarrow v$
11:   end for
12: until Maximum number of paths achieved.
```

training dataset. The sampling algorithm is illustrated in Algorithm 1. The sampled path is in the form of $\{e_0, r_0, e_1, r_1, ..., r_{T-1}, e_T\}$ where $e_t \in \mathcal{E}$ and $r_t \in \mathcal{R}$. Each consecutive triplet (e_t, r_t, e_{t+1}) is a fact stored in $\mathcal{G}$. Such a path provides an example to our graph generator **G** about what a possible reasoning path looks like.

We employ GPT-2, a pre-trained language model as the backbone of our graph generator **G**. Unlike COMET (Bosselut et al., 2019) which trains GPT with independent triplets, we trains GPT-2 with consecutive triplets as paths. We convert each sampled path to their text form $\mathbf{x} = \{X_0, Y_0, X_1, Y_1, ..., Y_{T-1}, X_T\}$, where $X_t = \{x_t^0, x_t^1, ..., x_t^{|e_t|}\}$ is the phrase tokens for the entity e_t and $Y_t = \{y_t^0, y_t^1, ..., y_t^{|r_t|}\}$ is the phrase tokens for the relation r_t. In order to further mimic the scenario where the model is provided with one question entity and one answer entity, here we add the last entity phrase tokens X_T together with a separate token *SEP* at the beginning of each path. By doing so, the generator would know where the final entity it should arrive at the end when generating the path. Since we would like to maximize the probability of such observed path, we use negative log likelihood as the loss function:

$$\mathcal{L} = -\sum_{\mathbf{x}} \log P(\mathbf{x}), \tag{1}$$

where $P(\mathbf{x})$ is the product of conditional probabilities:

$$P(\mathbf{x}) = \prod_{i=0}^{i=|\mathbf{x}|} P(x_i \mid x_{<i}), \tag{2}$$

The conditional probability is defined as:

$$P(x_i \mid x_{<i}) = \text{softmax}(h_t W). \tag{3}$$

Here h_t denote the final representation from GPT-2 for x_i and W is the embedding matrix for the vocabulary of both entity and relation phrases.

The benefits brought by learning to generate the sampled paths from $\mathcal{G}$ are two-fold. One is that we enrich the language model with structured knowledge such that it could generate paths with "commonsense" style for further reasoning. Another is that the unstructured knowledge encoded in the language model could alleviate the sparsity issue in KG. Some of the training paths are illustated in Table 1.

3.2 Fine-tuning on Task Dataset

Our ultimate goal is to empower the QA system with the generated reasoning paths as external evidence for better performance. In general, our whole framework for solving commonsense QA consists of two major parts. The first part is the aforementioned paths generator **G**. The second part is the classifier which is provided with both the structured evidence from the path generator and the unstructured one from a language model as input, and outputs the plausibility for each choice.

To encode the reasoning paths, we employ two different strategies which are called *offline* and *online* encoding respectively.

Offline Encoding

The strategy to encode the reasoning paths would leverage our generator **G** in a offline fashion. That being said, we take the generated paths from **G** as fixed discrete evidence to a path encoder, and just drop **G** away. Here, we use LSTM (Hochreiter and Schmidhuber, 1997) as our single path encoder and a multiplicative attention network (Luong et al., 2015) to aggregate multiple paths in a meaningful way.

For each pair of question entity e_i^q and choice entity e_j^a, the path generator **G** output a reasoning path p_k connecting them. Then p_k is fed to the LSTM encoder to get a single path embedding $\mathbf{p}_k$ with pooling over all hidden states. Since not all of the paths would contribute to the decision about which choice is the right answer, we employ an attention network to select softly the meaningful paths:

$$\mathbf{p} = \sum_k \alpha_k \mathbf{p}_k. \tag{4}$$

The attention weight α_k of each path embedding $\mathbf{p}_k$ is computed by

$$\alpha_k = \frac{\exp(s_k)}{\sum_{k'} \exp(s_{k'})}, \tag{5}$$

where

$$s_k = \mathbf{c}^\top \tanh(\mathbf{W}_a \mathbf{p}_k). \tag{6}$$

Here, the attention network is parametrized by a linear projection $\mathbf{W}_a$. The context embedding **c** is obtained by encoding the unstructured evidence via a language model. In specific, we simply concatenate the question and the choice with a separate token in between, namely $[q, \text{SEP}, a]$. Then we feed it to the language model to obtain its context embedding. In essence, the context embedding provides a guidance about which path is meaningful for solving the current question.

Online Encoding

Since the offline encoding of reasoning paths prevents us from training the whole framework end-to-end since the gradient could be back propagated through the discrete paths. Thus, we also consider employing an online encoding for representing the paths in a continuous way. In specific, we conduct pooling over the hidden states $\{h_t\}$ of the generator to get a path embedding $\mathbf{p}_k$ instead of leveraging LSTM. Then everything else remains the same design as in the offline encoding. Several advantages over offline encoding are brought by online encoding.

Table 2: Statistics on Commonsense QA.

Dataset	#Train	#Dev	#Test
Commonsense QA	8767	974	1221

1. We save the trouble of training the LSTM encoder which introduces additional parameters to our framework.
2. The generator is adjusted to the task dataset, which certainly bridge the discrepancy between pre-training and fine-tuning.
3. The hidden states from the generator surves a better function for encoding reasoning paths since the generator is pre-trained on both large corpora and structured triplet paths.

Fusing Heterogeneous Evidence

With unstructured evidence provided by context embedding **c** and structured one provided by paths embedding **p** at hand, our classifier leverages both of them to compute the plausibility of a question-choice pair. In specific, we concatenate **c** with **p** and feed them to a one layer of linear transformation to get a score for each question choice pair. Then the score is normalized by a softmax layer to get the final probability. The model is optimized by minimizing the cross-entropy loss.

4 Experimental Setup

We evaluate our method on the *CommonsenseQA* (Talmor et al., 2018), a multi-choice QA dataset evaluating a model's ability on reasoning with commonsense knowledge. We use the official develop set as our test set since the labels for the official test set are not released. We further randomly sample 10% of the official training set as our development set. The statistics for the dataset is shown in Table 2.

4.1 Dataset Processing

To extract all the entities for all the question sentences, we use plain string matching as in the previous work from Lin et al.. As for the entities for the answer choices, we simply treat each of them as a single entity since most of them are independent concepts in ConceptNet.

4.2 KG and Paths Sampling

We use ConceptNet as our commonsense KG. We discard all the triplets with some uninformative relation types [2] which offer little help for answering the questions.

When sampling paths from ConceptNet, we adopt two different strategies as follows.

1. Local Sampling. All the random walks start from the entities which appear in the questions of the task training set. This setting help our generator to generate paths which are more task-dependent.

2. Global Sampling. We also randomly pick some entities from KG and conduct random walks starting from them. This would help increase the diversity of our paths dataset and prevent our generator biased towards local structure of KG.

For both of the strategies, we sample paths with hops ranging from 3 to 5 in order to construct paths dataset with mixing hops. This would help our generator learn to connect entities with different hops of paths as needed. We finally obtain 672378 paths in total and split them into training/develop/test set with ratio of $8:1:1$.

Since the backbone of our generator is a pre-trained language model, we convert each relation in KG into mention phrases with predefined templates. This is crucial for us to leverage the knowledge encoded in the language model. The resulting paths would be of more "natural language" style, with which the language model could be thus friendly triggered to generate paths.

4.3 Hyper-parameters

We employ a pre-trained GPT2-base model (Radford et al., 2019) as the initialization of our generator. Then we fine-tune the generator with a initial learning rate of $1e-5$ and a batch size of 128. The learning rate is changed with a warm-up period of 1000 mini batches and then linearly decayed. The training lasts until the loss on development set no longer decreases for 3 epochs.

For training over task datasets, we search the optimal hyper-parameters based on the classification accuracy on development set. The initial learning rate is choosing from $\{5e-5, 1e-5, 5e-6, 1e-6\}$. The batch size is choosing from $\{8, 16, 32, 64, 128\}$. Large batch size is achieved by accumulating gradient through several small batches. We also train our model with a warm-up period of 500 mini-batches and linearly decrease the learning rate. The training lasts until the accuracy no longer increases for 5 epochs.

Table 3: Classification Accuracy on Commonsense QA.

Models	Dev (%)
RoBERTa	72.89
KagNet	64.46
OCN + CN	67.30
RoBERTa + CSPT	76.2
Our (Online)	72.56

4.4 Baselines

We consider baselines including pure language models and other methods which leverage structured knowledge and/or unstructured knowledge.

Pre-trained Language Model. Since our goal is to enhance the QA system with external knowledge and part of our model relies on pre-trained language model, we compare our method to a strong baselines, i.e., RoBERTa (Liu et al., 2019). As in our framework, we use the *CLS* token embedding from RoBERTa as the context embedding and feed it to a linear classifier to obtain the score. We fine tune RoBERTa with a learning rate of $1e-5$ and batch size of 32 as suggested by previous works.

Models with Static KG. Another contribution of our work is that our model builds dynamically KG for commonsense QA. Therefore, we compare our method with previous ones which rely on static KG. KagNet (Lin et al., 2019) adopts the same task setting as ours except that the subgraph is located by finding all paths connecting question entities and answer entitites on existing ConceptNet. OCN+CN (Ma et al., 2019) also adopts a similar setting, but they only consider one-hop relations between questions entities and answers entities.

Model with both KG and Text. We also consider RoBERTa+CSPT, a model which pre-trained over synthetic dataset constructed from ConceptNet and Open Mind Common Sense (OMCS) corpus (Singh et al., 2002).

[2]Including *relatedto*, *synonym*, *antonym*, *derivedfrom*, *formof*, *etymologicallyderivedfrom*, and *etymologicallyrelatedto*

Table 4: Effect of Joint Training. Classification Accuracy on Commonsense QA In-house (Ih) Develop Set and Official Develop Set (as Test Set).

Models	Ih Dev (%)	Test (%)
RoBERTa (base)	53.23	54.14
Offline	43.08	44.80
Online	**54.56**	**54.96**

5 Experimental Results

5.1 Overall Performance

The general results on Commonsense QA is shown in Table 3. We could observe that compared with models using static KG, our model performs much better with an accuracy increase as much as 8.10%. This demonstrates the effectiveness of our model in generating contextual KG which is helpful in addressing commonsense QA. One drawback of our model is that when it comes to incorporating external knowledge to pretrained language model, our method does not bring any improvement. This could be due to the simple concatenation of context embedding and path embedding, which is not the optimal way to infuse structured and unstructured knowledge. Also, our model is outperformed by RoBERTa+CSPT by a considerable margin (3.64%). We believe its performance gain is brought by fine-tuning RoBERTa on the OMCS corpus, which helps the pretrained language model adapts to Commonsense QA dataset more easily.

5.2 Effect of Joint Training

The results in Table 4 show that our online model outperforms both fine-tuned RoBERTa (base) and especially our offline model with a significant margin. This demonstrates the effectiveness of joint training of our whole framework, including path generator and reasoner. During joint training, both the generated paths and their embeddings are adjusted to the task dataset, which is crucial for transferring knowledge from KG to the QA system. On the other hand, our offline model performs poorly compared with RoBERTa. Note that, this does not mean that the generated paths are not helpful in assisting the QA system. It could be the case that the LSTM path encoder introduces more parameters to the whole model and are not well-learned due to limited supervision signal.

5.3 Robustness to Sparsity

One of the biggest advantages of providing reasoning paths to QA system is that these external evidence introduces more inductive bias to the model. This would greatly help the model to be survive from less training data and generalize better. Therefore, we randomly sample $\{20\%, 40\%, 60\%, 80\%, 100\%\}$ of the training data to see whether our model is more robust than the baseline. Results in Table 5 does show a consistent performance across different amount of training data. On one hand, our model does outperform fine-tuned RoBERTa when 60% and 100% of the training data is used. However, the gap between them is marginal. This also expose the problem of our model failing to efficiently combine the evidence from two perspectives. On the other hand, when the training data becomes sparser, fine-tuned RoBERTa continues to outperform our model and the performance gap is increasing larger. This demonstrates the robustness of large language models. Indeed, the performance of fine-tuned RoBERTa does not drop significantly when less training data is used. We guess that the commonsense knowledge which is necessary to solve those questions are already stored in these large-capacity language models. Even with less training data, these language models still manage to reorganize its information to serve the downstream tasks.

6 Related Work

Both structured knowledge stored in a knowledge base and unstructured text from large web corpus proved to be valuable sources for question answering. According to how the knowledge from both sizes cooperates, we list out three categories of these works which 1) supplement inference over KB with text, 2) supplement inference over text with KB, or 3) fusing knowledge from KB and text jointly.

6.1 Text to KB

In order to address the incompleteness of KB, several works augment their models with external evidence from text data. For some of them, text is only used as additional feature to enhance the inference over KB (Krishnamurthy and Mitchell, 2012; Reddy et al., 2014; Choi et al., 2015; Savenkov and Agichtein, 2016; Lin et al., 2019). They utilize external text to better under-

Table 5: Classification accuracy on test set with models trained on different amount of training set. Results are obtained by running experiments for 4 times with different random seed.

Model	20%	40%	60%	80%	100%
RoBERTa Fine-Tuned	67.16(±0.90)	69.75(±1.14)	69.59(±1.20)	72.25(±1.10)	72.53(±1.05)
Our (Online)	64.60(±1.00)	68.22(±0.88)	70.72(±0.79)	72.24(±0.33)	72.59(±0.57)

stand the questions as well as enrich the features for candidate answers. Recent work (Fu et al., 2019) also make use of corpus for extracting new facts to complete KB during inference.

As methoned above, these methods are better at compositional reasoning over KB which unstructured text do not support (Das et al., 2017), and are greatly improved when enhanced by text evidence. However, when faced with more open-domain questions, evidence from text might be more useful and should serve as the main contribution instead of a complementing role. Moreover, they neglect the other side where structured knowledge could help inference over text.

6.2 KB to Text

There also exist some works investigating the reverse direction, i.e., leveraging KB to improve inference over text. For example, the work in Sun et al. 2015 links each candidate answer in search text to the entities in KB in order to get their semantic feature. Further, Xiong et al. 2019 employs gating mechanism to incorporate necessary strucutred knowledge to better encode questions and passages. While these methods make up the shortage of KB-oriented counterparts, the obvious limitation is that the factoid knowledge in KB is not consulted to obtain answers directly. Likewise, they also omit the possible benefits brought by the text-to-KB line of approaches.

6.3 Fusing KB and Text

Limited attention is drawn to exploit evidence from KB and text jointly for integral reasoning. Early works utilizing both sources adopt a late fusion strategy. They either aggregate predictions which are grounded independently from each size (Ferrucci et al., 2010; Baudiš, 2015), or simply unify structured and unstructured knowledge with universal schema and feed them to memory network as input (Das et al., 2017). As pointed out by Sun et al. 2018, this strategy is sub-optimal, as models have limited ability to aggregate evidence across the different sources and ignore the rich inter relations between both sizes. To bridge these gaps, Sun et al. 2018 adopts an early fusion strategy instead. They firstly construct a question subgraph to incorporate both KB and corpus via entity links. Then they propose heterogeneous update rules to fuse knowledge from different nodes. Lv et al. 2019 adopts a similar strategy to construct a graph from both sources but their method to fuse the heterogeneous knowledge is twisted. Firstly, nodes from both sizes are presorted as sequences and concatenated into one single input of a language model which generates a sequence representation. Then graph neural networks are used to generate representation for the whole graph. Finally, both the sequence and graph representations are used to compute the prediction score.

To some extent, these works step further to exploit both KB and text in a more unified way than the works introduced in the previous two subsections do. Therefore, evidence from both sizes could be considered jointly to better answer the question of any kind. Still, they emphasize more on relying question to select useful evidence from KB and text. The interaction of both sizes is fulfill only by knowledge fusion. Possible guidance from one size to encode the other size is not explicitly investigated.

7 Conclusion and Future Works

We propose to learn a reasoning paths generator aiming to provide external evidence dynamically to assist QA system. Expermental results show that our current method does not improve much over fine-tuned language models, exposting two major problems or directions for us to develop our method. One is that we have no gurantee on whether the generated paths could really help better reasoning, and we also lack the ground truth for useful paths. Probably we need to exploit more features from the given context to help us select paths. Another is that our current way of incorporating structured and unstructured evidence is not optimal in fusing heterogeneous information. A better design for increasing interaction between the two modules is needed.

References

Petr Baudiš. 2015. Yodaqa: a modular question answering system pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering*, pages 1156–1165.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317*.

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

Eunsol Choi, Tom Kwiatkowski, and Luke Zettlemoyer. 2015. Scalable semantic parsing with partial ontologies. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1311–1320.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. *arXiv preprint arXiv:1704.08384*.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79.

Cong Fu, Tong Chen, Meng Qu, Woojeong Jin, and Xiang Ren. 2019. Collaborative policy learning for open knowledge graph reasoning. *arXiv preprint arXiv:1909.00230*.

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

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos qa: Machine reading comprehension with contextual commonsense reasoning. *arXiv preprint arXiv:1909.00277*.

Jayant Krishnamurthy and Tom M Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 754–765. Association for Computational Linguistics.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455.

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

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

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

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*.

Kaixin Ma, Jonathan Francis, Quanyang Lu, Eric Nyberg, and Alessandro Oltramari. 2019. Towards generalizable neuro-symbolic systems for commonsense question answering. *arXiv preprint arXiv:1910.14087*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019a. Atomic: an atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019b. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Denis Savenkov and Eugene Agichtein. 2016. When a knowledge base is not enough: Question answering over knowledge bases with external text data. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 235–244. ACM.

Push Singh, Thomas Lin, Erik T Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1223–1237. 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*.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. *arXiv preprint arXiv:1809.00782*.

Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. 2015. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1045–1055. International World Wide Web Conferences Steering Committee.

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

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2019. Improving question answering over incomplete kbs with knowledge-aware reader. *arXiv preprint arXiv:1905.07098*.