Learning to Generate Multi-Hop Knowledge Paths for Commonsense Question Answering

Anonymous authors

Abstract

Commonsense question answering (QA) requires a model of general background knowledge about how the world operates and how people interact with each other before reasoning. Prior works focus primarily on manually curated commonsense knowledge graphs, but these knowledge graphs are incomplete and thus may not contain the necessary knowledge for answering the questions. In this paper, we propose to learn a multi-hop knowledge path generator to generate structured evidence dynamically according to the questions. Our generator uses a pre-trained language model as the backbone, leveraging a large amount of unstructured knowledge stored in the language model to supplement the incompleteness of the knowledge base. The experiments on two commonsense QA datasets demonstrate the effectiveness of our method, which improves over strong baselines and also provides human interpretable explanations for the predictions.

1. Introduction

Answering commonsense questions requires background knowledge about how the world operates (e.g., intuitive physical commonsense) and how people interact with each other in daily life (e.g., intuitive psychological commonsense). For example, the answer to a multi-choice question "In what geological feature will you find fungus growing?" is "cave". To answer this question, a question answering (QA) system needs commonsense like fungus often grows in a moist and warm environment, which could be provided by a cave and cave happens to be a kind of geological feature. Such knowledge is obvious for humans but not easy for most of the existing QA systems to possess [Talmor et al., 2018].

Recent advances in pre-trained language models have brought great successes to many natural language understanding (NLU) tasks [Radford et al., 2018, Devlin et al., 2018], with impressive results on commonsense-related benchmarks [Zellers et al., 2018, Bhagavatula et al., 2019, Huang et al., 2019]. However, it is unclear whether they indeed achieved commonsense reasoning or just captured correlations in the datasets [Niven and Kao, 2019]. An approach to address this problem is to leverage the commonsense knowledge graphs (KG) such as ConceptNet [Speer et al., 2017] or ATOMIC [Sap et al., 2019] to provide additional background knowledge. A benefit of leveraging such KGs is also to add interpretability to the model. The typical approach to leverage these commonsense KGs for QA is to retrieve a local graph of the entities mentioned in the questions and answer choices from a static KG and then conduct reasoning over the local graph for predicting the answer. However, leveraging commonsense KGs is difficult, posing the following challenges. (1) **Noise**: While a commonsense KG contains rich information, only a few facts in the local subgraph that connect the entities associated with question/answer pairs will be informative for addressing the tasks; many connections in the local graph are noisy, unhelpful signals for learning. (2) **Sparsity**: KGs are also known to be incomplete [Li et al., 2016]; the facts necessary for answering the questions are often missing from a hand-crafted KG.

To address these challenges, we propose a path generator based on a pre-trained language model that learns to generate multi-hop knowledge paths as a dynamic knowledge graph. These generated paths can efficiently include relevant information for answering commonsense questions while excluding noises. Moreover, the large amounts of knowledge encoded in pre-trained language models provides our path generator with better generalizability to combat the sparsity of knowledge graphs.

In our approach, we first sample a set of random walk instances from a static commonsense KG with manual heuristics (Section 3.1) to ensure their informativeness and helpfulness. Then we fine-tune a pre-trained language model, GPT-2 [Radford et al., 2019] on the sampled paths to learn a knowledge path generator (Section 3.2). Given any pair of question and answer entities, our path generator dynamically generates a novel multi-hop path to connect them instead of retrieving the path from a static KG. These generated paths further serve as the local graph for our KG augmented QA system to solve the commonsense questions (Section 3.3).

We conduct experiments on two benchmark datasets *CommonsenseQA* [Talmor et al., 2018] and *OpenBookQA* [Mihaylov et al., 2018]. The results show that our generator can efficiently generate knowledge paths that help improve the performance over strong baselines, while providing interpretable explanations. Ablation studies demonstrate that the effectiveness is due to a large extent from our strategies for learning the path generator.

2. Problem Statement

In this paper, we focus on learning a graph generator that generates a local knowledge graph dynamically for each question, such that the local graph (1) is informative for answering the question and (2) contains knowledge that are missing from the sparse KG. We reduce the local graph generation problem to multi-hop knowledge paths generation problem. Assume we employ a reasonable entity recognition system (in practice we simply use string matching) to extract all the entity mentions in the question $\{e_i^q\}$ and the answer choices $\{e_j^a\}$. Given a pair of question entity and choice entity, our path generator learns to output a knowledge path as a snippet of the commonsense KG that connects the two entities. We hypothesize that the generated knowledge paths contain crucial information to facilitate the reasoning process of the QA systems.

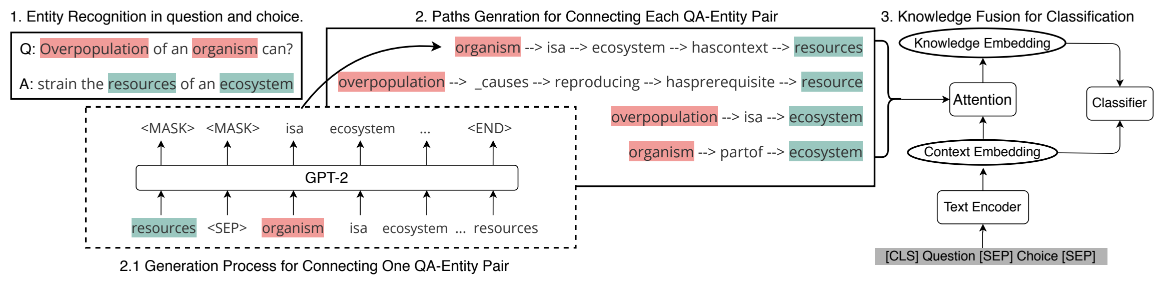

1. Entity Recognition in question and choice.
Q: Overpopulation of an organism can?
A: strain the resources of an ecosystem
2. Paths Genration for Connecting Each QA-Entity Pair
organism --> isa --> ecosystem --> hascontext --> resources
overpopulation --> _causes --> reproducing --> hasprerequisite --> resource
overpopulation --> isa --> ecosystem
organism --> partof --> ecosystem
<MASK> <MASK> isa ecosystem ... <END>
GPT-2
resources <SEP> organism isa ecosystem ... resources
2.1 Generation Process for Connecting One QA-Entity Pair
3. Knowledge Fusion for Classification
Knowledge Embedding
Attention
Classifier
Context Embedding
Text Encoder
[CLS] Question [SEP] Choice [SEP]

Figure 1: **Overview of the Proposed Approach.** (1) extract entities from questions and answer choices, (2) use our path generator to generate a multi-hop knowledge path to connect each pair of question and answer entities, (3) aggregate the generated paths as a knowledge embedding, and fuse it with a context embedding from a text encoder for classification.

3. Multi-Hop Knowledge Path Generator as Dynamic KG

We propose to learn a knowledge path generator as a dynamic knowledge graph, which generates relevant knowledge paths to connect the entities in the question the answers to help QA systems answer commonsense questions. We compose training data for the generator by sampling a set of knowledge paths from a commonsense KG via random walk with heuristics for informative paths (Section 3.1). We then fine-tune a pre-trained language model - GPT-2 as our path generator. (Section 3.2). Finally, we integrate the knowledge path generator with our reasoning system (Section 3.3). This system uses another pre-trained language model to encode the question-choice pair to produce a textual embedding. Such a textual embedding serves at first as the attention query to aggregate the relational paths from our generator to obtain the knowledge embedding. Then the textual embedding together with the knowledge embedding are used as the question-choice representation for the final classification. Figure 1 illustrates the pipeline of our method.

3.1 Knowledge Paths Sampling

We sample paths from a commonsense KG using random walk to provide representative relational paths as training data for our knowledge path generator. Given a static KG $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ is the entity set and $\mathcal{R}$ is the relation set, a sampled path is a random walk on the graph taking the form of $\{e_0, r_0, e_1, r_1, ..., r_{T-1}, e_T\}$, where $e_t \in \mathcal{E}$ and $r_t \in \mathcal{R}$. T is the number of hops which is a hyperparameter. We assume such paths contain relevant knowledge for the commonsense QA tasks. To improve the quality of the paths, we adopt two heuristic strategies. For relevance, we define a subset of relation types that are useful for a specific commonsense question dataset, and we filter out all the irrelevant relation edges in the commonsense KG before we conduct sampling. For informativeness, we require that each path should not contain edges with repeated relation types. We select the starting node of the random walks based on the following two sampling strategies:

1. **Local Sampling**. The random walks start from the entities which appear in the questions and answer choices of the task training set. This strategy helps our generator to generate paths which are more task-centered.

Table 1: **Transformation of Sampled Paths to Input for Generator.** During inference, the target entity, [SEP] token and starting entity (in grey) are given to our generator to generate the path which is supposed to join the two entities.

{predator,distinctfrom,prey,isa,animal}
→ { animal,[SEP],predator ,distinctfrom,prey,isa,animal}

{eat_lot,_hasprerequisite,get_fat,_hassubevent,sitting_quietly}
→ { sitting_quietly [SEP],eat_lot ,_hasprerequisite,get_fat,_hassubevent,sitting_quietly}

2. **Global Sampling**. We also randomly sample some entities from KG and conduct random walks starting from them. This helps increasing the diversity of the sampled paths and prevent biasing our generator towards the local structure of KG.

We add a reverse relation r^{-1} for each relation r so that a sampled path can contain reverse triplets, e.g., (o, r^{-1}, s). We also sample paths with a mixed number of hops (T) to train our generator to connect entities using paths with variable lengths when needed. The algorithm is illustrated in Algorithm 1 in the appendix.

3.2 Generating Paths to Connect Entities

In order to learn a relational paths generator as a neural KG to overcome the sparsity issue of a static KG, we employ GPT-2 as the backbone of the generator. GPT-2 is a pre-trained language model of large capacity which encodes rich unstructured knowledge from a huge corpus. The benefits brought by doing so are two-fold. One is that we enrich the language model with structured knowledge such that it could generate paths with "commonsense" style as designed. Another is that the unstructured knowledge encoded in the language model could alleviate the sparsity issue in KG.

Unlike COMET [Bosselut et al., 2019] which fine-tunes GPT (an earlier version of GPT-2) with independent triplets, we fine-tune GPT-2 with consecutive triplets as paths which are obtained in Section 3.1. To do so, we first convert each symbolic path to their textual form $\mathbf{x} = \{X_0, Y_0, X_1, Y_1, ..., Y_{T-1}, X_T\}$, where $X_t = \{x_t^0, x_t^1, ..., x_t^{|e_t|}\}$ are the phrase tokens of the entity e_t and $Y_t = \{y_t^0, y_t^1, ..., y_t^{|r_t|}\}$ are the phrase tokens of the relation r_t. This is crucial for leveraging the knowledge encoded in the language model. The resulting paths mimic natural language making them usable in language model. In order to further mimic the scenario where the generator is provided with one question entity and one choice entity, here we add the last entity phrase tokens X_T together with a separate token [SEP] at the beginning of each path (as illustrated in Table 1). By doing so, the generator will be aware of the last entity it should output when generating a path. Since we would like to maximize the probability of such observed paths given the entity pair, we use negative log likelihood as the loss function:

$$\mathcal{L} = -\sum_{\mathbf{x}} \log P(\mathbf{x}|X_T, [SEP], X_0), \qquad (1)$$

where $P(\mathbf{x}|X_T, [SEP], X_0)$ is the product of conditional probabilities:

$$P(\mathbf{x}|X_T, [SEP], X_0) = \prod_{t=|X_0|+|X_T|+1}^{|\mathbf{x}|} P(x_t \mid x_{<t}), \tag{2}$$

The conditional probability is defined as:

$$P(x_t \mid x_{<t}) = \text{softmax}(\mathbf{W}_{vocab} \cdot \mathbf{h_t}). \tag{3}$$

Here $\mathbf{h_t}$ denotes the final representation from GPT-2 for x_t and $\mathbf{W}_{vocab}$ is the embedding matrix for the vocabulary used by GPT-2.

3.3 Commonsense QA System with Relational Paths Generator

Our ultimate goal is to include in the QA system the generated reasoning paths as external evidence for better performance and interpretability. In general, our framework for solving commonsense QA consists of two major parts. The first part is the aforementioned path generator. The second part is a contextual encoder which encodes the question and choice to output a context embedding $\mathbf{c}$ as **unstructured evidence**. In this paper, we employ the bidirectional transformers style pre-trained language model [Devlin et al., 2018, Liu et al., 2019], a commonly used contextual encoder for textual input. The question and choice are concatenated with some special tokens added, and then fed to the contextual encoder to obtain $\mathbf{c}$. The context embedding is further used as guidance for representing the paths generated by the paths generator, which outputs a knowledge embedding $\mathbf{p}$ as **structured evidence**. Finally, these two types of evidence are fed to a classifier to output a plausibility score for each choice. We depict the knowledge embedding module as follows.

Knowledge Embedding as Structured Evidence For each pair of question entity e_i^q and choice entity e_j^a, the path generator outputs a reasoning path p_k connecting them as it has been fine-tuned to. To represent these discrete paths, we maximize the usage of our generator by taking the average of the last layer hidden states from GPT-2 (before the softmax layer in Eq. 3) as our path embedding, i.e.,

$$\mathbf{p_k} = \text{MEAN}(\{\mathbf{h_0}, \mathbf{h_1}, ..., \mathbf{h_T}\}). \tag{4}$$

Since GPT-2 has been pre-trained on a large corpus, we believe such representation should be sufficient in preserving the information of the paths. Moreover, this saves us the trouble of learning an additional path encoder.

Since not all of the paths would contribute equally to the decision about which choice is the right answer, we leverage the unstructured evidence, i.e., the context embedding $\mathbf{c}$ mentioned above as the guidance on encoding this structured evidence. Specifically, we extend Relational Network (RN) [Santoro et al., 2017] with attention mechanism to select the meaningful paths softly:

$$\mathbf{p} = W_{proj} \cdot \sum_k \alpha_k \mathbf{p}_k, \tag{5}$$

where W_{proj} is a learnable projection matrix. The attention weight α_k of each path embedding $\mathbf{p}_k$ is computed by

$$\alpha_k = \frac{\exp(s_k)}{\sum_{k'} \exp(s_{k'})}, \tag{6}$$

where

$$s_k = \mathbf{c}^\top \tanh(\mathbf{W}_{att} \cdot \mathbf{p}_k + \mathbf{b}_{att}). \tag{7}$$

Here, the attention network is parametrized by ($\mathbf{W}_{att}$,$\mathbf{b}_{att}$).

Fusion of Heterogeneous Evidence for Classificaion 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. We concatenate **c** with **p** and feed them to the final classification layer, which is a linear transformation to get a score for each question choice pair $\{q, a\}$:

$$f(q, a) = \mathbf{W}_{cls} \cdot [\mathbf{c}; \mathbf{p}] + \mathbf{b}_{cls}, \tag{8}$$

where the linear classification layer is parameterized by $(\mathbf{W}_{cls}, \mathbf{b}_{cls})$. Then the score is normalized by a softmax layer to get the final probability over all choices. The model is optimized by minimizing the cross-entropy loss. Learnable parameters include all the modules described above excluding our proposed path generator since during experiments we find that fixing the generator yields better performance. This also reflects another advantage of our path generator: after being fine-tuned on sampled random walks from KG, the path generator could be used as a plug-in module to an existing QA system and needs no further training.

4. Experiments

4.1 Experimental Setup

We evaluate our method on the *CommonsenseQA* [Talmor et al., 2018] and *OpenBookQA* [Mihaylov et al., 2018], which are both multi-choice QA datasets evaluating a model's ability to reason with commonsense knowledge. For *CommonsenseQA*, we use the official development 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. For *OpenBookQA*, we do not use the additional set of science facts originally provided by the dataset and rely on our generator to provide background knowledge. More information about the datasets could be found in Appendix B.

4.2 Dataset Processing

KG and Entity Recognition We employ ConceptNet [Speer et al., 2017] as our commonsense KG due to its broad coverage of general background knowledge about the world. As mentioned in Section 3.1, we discard all the triplets with the predefined uninformative relations (see Appendix C) before paths sampling.

To extract all the entities mentioned in the question and answer choices, we use plain string matching as in the previous work from Lin et al.. One exception is that for the answer choices in *CommonsenseQA*, we treat each of them as a single entity since most of them are independent concepts in ConceptNet.

Paths Sampling We sample paths with hops ranging from 1 to 3 to construct the set of paths with a mixed number of hops. The number of paths which we obtain from both global sampling and local sampling on specific task datasets is shown in Appendix B. We combine the paths from the two sampling strategies and further split them into training/development/test set with ratio of $9 : 0.5 : 0.5$.

4.3 Baselines

We consider several baselines including fine-tuned language models and KG-augmented models with static KG. We also propose a baseline which conducts link prediction between questions and answers entities as a 1-hop neural KG.

Pre-trained Language Model. Since our goal is to enhance the QA system with external knowledge and part of our model relies on the pre-trained language model, we compare our method to two strong baselines, i.e., BERT [Devlin et al., 2018] and RoBERTa [Liu et al., 2019] and call them as **Fine-tuned LM**. As in our framework, we use the mean pooling of the last layer of hidden states from RoBERTa as the context embedding and feed it to a linear classifier to obtain the score. We fine-tune these language models with the hyperparameters suggested by previous works.

Models with Static KG. Since we argue that our dynamic neural KG is superior, we compare our method with previous works with different graph encoders for modeling local graphs retrieved from the static KG. Firstly, we compare with a degenerate version of our method, i.e., **RN** with attention mechanism. Other advanced baselines include **KagNet** [Lin et al., 2019], Relational Graph Convolutional Networks **(RGCN)** [Schlichtkrull et al., 2018], **GconAttn** [Wang et al., 2019] and KV-Memory **(KVM)** [Mihaylov and Frank, 2018]. We concatenate the knowledge embedding from each of these KG-augmented methods with the context embedding from the pre-trained language model as the input to the classification layer (Eq. 8) for a fair comparison.

Model with Link Prediction. We also propose a baseline model called **Link Prediction**, which predicts the relation between question and answer entities instead of generating the knowledge paths. We first employ TransE [Bordes et al., 2013] to learn a knowledge representation for each entity and relation in ConceptNet. Then for each pair of question and answer entities, we predict their 1-hop relation based on their knowledge representation. Then for each resulting triplet, we concatenate their knowledge representations as a 1-hop path embedding. The remaining module design is the same as our method.

4.4 Overall Results

The results on *CommonsenseQA* and *OpenBookQA* are shown in Table 2. When equipped with RoBERTa as the context encoder, our method (Path Generator in the table) outperforms all baselines. On *OpenBookQA*, our method achieves close to 2% accuracy gain over the second best baseline, indicating the effectiveness of the generated paths as the structured evidence. Also with RoBERTa, all the models with static KG fail to outperform fine-tuned LM on both datasets, which demonstrates the superiority of the dynamic KG built by our path generator. Such superiority is also shown by the improvement of our method over the Link Prediction baseline. This baseline outperforms several methods with static KGs on both datasets, indicating that even predicting 1-hop knowledge paths is helpful to address the sparsity issue. By providing multi-hop knowledge paths which are more informative, our main method outperforms the Link Prediction baseline considerably. When equipped with BERT as the context encoder, our method does not outperform all the models with

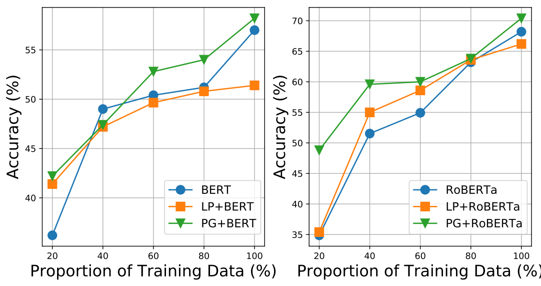
Table 2: **Classification Accuracy on *CommonsenseQA* and *OpenBookQA* datasets.** Results are taken from 4 runs of experiments with different random seeds, excluding outliers. The result for KagNet is reported as development performance from the paper (top score in boldface, second score underlined).

Methods	*CommonsenseQA*		*OpenBookQA*	
	BERT-large	RoBERTa-large	BERT-large	RoBERTa-large
Fine-tuned LM	60.36 (±0.71)	72.53 (±1.05)	57.00 (±1.91)	<u>68.21</u> (±0.88)
RN	**63.36** (±0.26)	<u>73.65</u> (±3.09)	57.16 (±2.14)	65.20 (±1.18)
KagNet	62.35	-	-	-
RGCN	61.78 (±1.49)	71.82 (±1.92)	**58.30** (±1.80)	62.45 (±1.57)
GconAttn	<u>62.77</u> (±1.35)	73.23 (±1.22)	57.75 (±0.55)	64.75 (±1.48)
KVM	61.19 (±0.81)	73.52 (±0.79)	58.10 (±0.94)	67.40 (±0.75)
Link Prediction+RN	60.44 (±0.62)	72.89 (±1.03)	51.43 (±0.00)	66.29 (±0.47)
Path Generator+RN	61.67 (±0.79)	**74.01** (±0.62)	<u>58.22</u> (±1.74)	**70.47** (±0.53)

Table 3: Ablation Study: Different Variants of Our Method.

Setting	*CommonsenseQA*
Gen-Random	72.89 (±0.52)
Gen-Local	72.62 (±0.72)
Gen-Global	73.22 (±0.94)
Full	74.01 (±0.62)

Figure 2: Accuracy on *OpenBookQA* with different proportion of training data.



static KGs. This is the limitation of our method in the sense that it still relies on the context encoder to aggregate the paths with attention mechanism to some extent. How to design a paths generator which is less coupled with the textual module is future work.

4.5 Performance Analysis

Ablation Study We conduct further analyses to study the contribution of different strategies for learning our generator. We ablate our paths generator with each of the following training strategies. (1) **Gen-Random**. We train a randomly-initialized GPT-2 on the sampled paths as our paths generator instead of fine-tuneing a pre-trained one. (2) **Gen-Local/Global**. We fine-tuned a pre-trained GPT-2 on the sampled paths under either local or global sampling. The ablation results are displayed in Table 3. All variants of our path generator achieve worse performance than the full method. The worst performance comes from **Gen-Local**, which suggests that in order to learn a commonsense paths generator, it is necessary to feed it with broader knowledge. This is also validated by the performance of **Gen-Global** which gives less performance drop. It also shows that it is helpful to feed some paths centered around the task-specific entities to the generator. Although trained on

Table 4: Generated paths from question entities to gold answer entities.

Q1: Where would you find **magazines** along side many other printed works?
A: doctor. B^* : *bookstore*. C: market. D: train station. E: mortuary.
Path from Full: {magazine, isa, book, atlocation, bookstore} (2-hop)
Path from Random: {magazine, _isa, magazine, atlocation, bookstore}
Q2: If you want **harmony**, what is something you should try to do with the world?
A: take time. B. make noise. C. make war. D^*.*make peace*. E. make haste.
Path from Full: {harmony, _hassubevent, make better world, hasprerequisite, make peace} (2-hop)
Path from Random: {harmony, _usedfor, committing perjury, causes, make peace}
Q3: Janet was watching the **film** because she liked what?
A: rejection. B: laughter. C^* : *being entertained*. D: fear. E: bordem.
Path from Full: {film, usedfor, being entertained} (1-hop)
Path from Random: {film, _hascontext, being entertained}
Q4: What do people typically do while playing guitar?
A: cry. B: hear sounds. C^*: singing. D:arthritis. E:making music.
Path from Full: {guitar, usedfor, playing music, _causes, singing} (2-hop)
Path from Random: {guitar, hascontext, music, _causes, singing}

both global and local sampled paths, **Gen-Random** still achieves the second-worst performance. This demonstrates that learning the knowledge paths from scratch only provides the generator with what a static KG has already. The unstructured knowledge stored in a pre-trained GPT-2 helps to complement what a static KG might lack.

Performance under Limited Labeled Data We investigate the low-resource scenario where we only use {20%, 40%, 60%, 80%, 100%} of the training data from *OpenBookQA* for training to see whether our model is more robust to data sparsity than the baselines. The results from Figure 2 show that our method (with RoBERTa) outperforms or is comparable to the baselines with different amounts of training data. The performance gain is more considerable when extremely less data is used. We also notice a consistent behavior where the RoBERTa equipped models perform better than the BERT equipped ones, which indicates part of the robustness might also come from a stronger text encoder. Still, the superiority of our method and even the Link Prediction baseline demonstrates the effectiveness of introducing structured evidence as inductive bias for the low-resource setting.

4.6 Case Study on Model Interpretability

In Table 4, we show case studies on the paths generated respectively by our full method and the **Gen-Random** variant for connecting the question entities to the gold answer entities. The complete case studies are shown in Appendix E. In Q1, we observe that our path generator can provide knowledge about the location of the entity with a 2-hop path, which is helpful in answering this kind of "Where" question. Although the path from **Gen-Random** also contains the *atlocation* relation, it fails to predict properly for the first hop knowledge (*_isa*). In Q2, we observe that the path from our full model provides knowledge about the continuous intention of human with a 2-hop path while the path from **Gen-Random** contains wrong information by stating that *peace* is caused by *committing perjury*. In Q3, we find that the path from our full model could provide knowledge about the property of entity and also figure out when a 1-hop relation is sufficient to connect the question and answer entity. For **Gen-Random**, however, it fails to predict a more informative relation (*_hascontext*). Similar observation is also seen in Q4. All these cases demonstrate that a fine-tuned pre-trained GPT-2 is superior in generalizing commonsense paths by leveraging the usntructured knowledg encoded in it.

5. Related Work

Multi-hop Reasoning on KGs. Like commonsense QA, the recent benchmark datasets in the fields of open domain QA [Yang et al., 2018], reading comprehension [Welbl et al., 2018], etc., also require the corresponding systems to conduct multi-hop reasoning. Significant work exists to develop such models based on static KGs. Typically, these works employ entity linking systems to recognize the entities mentioned in the context and then retrieve the knowledge paths as the local graph structure around the entities. They further score or rank the retrieved paths using graph-based metrics (e,g., PageRank, centrality) [Paul and Frank, 2019, Fadnis et al., 2019], handcrafted rules [Kapanipathi et al., 2019] or neural methods (e.g., attention mechanisms) [Kundu et al., 2018, Lin et al., 2019]. The main difference between their work and ours is that rather than relying on a static KG, our paths generator is able to generate knowledge paths on the fly, which could be absent from an incomplete KG.

Dynamic Knowledge Path Generation or Prediction. Prior work also investigates methods generate or predict reasoning or knowledge paths instead of extracting them from some static KGs. Work by Asai et al. [2019] learns to predict evidence documents sequentially to form their reasoning paths, but still requires the inter-links between documents on their constructed KG. Fu et al. [2019] proposes a fact extractor for retrieving missing facts to complement the incomplete KGs. But they limit their setting to knowledge graph reasoning, where both a query entity and a single query relation are given. The most relevant work to ours is from Bosselut and Choi [2019] which also leverages the language model GPT-2 to dynamically generate knowledge paths. However, they expand their paths by predicting the next entity one at a time while we generate the paths in an end-to-end manner. Moreover, their method is limited to the setting where the whole context could be treated as a single entity and the question could be treated as a query relation. We do not have such a limitation and could be applicable to more general commonsense QA.

6. Conclusion

This paper proposes a generator which generates multi-hop knowledge paths as structured evidence for answering commonsense questions. To learn such a path generator, we fine-tuned GPT-2 on the random walks sampled from a commonsense KG. Then the generator connects each pair of question and answer entity with a knowledge path. These paths are further aggregated as knowledge embedding and fused with context embedding given by a text encoder for classification. Experimental results on two benchmark datasets demonstrate the effectiveness of our method in outperforming both strong pre-trained language models and static KG augmented methods. Besides the improvement, we also show that the generated paths are interpretable in terms of their informativeness and helpfulness. Future works include how to decouple the generator with the text encoder and a better way to fuse the knowledge.

References

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. Learning to retrieve reasoning paths over wikipedia graph for question answering. *arXiv preprint arXiv:1911.10470*, 2019.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. Abductive commonsense reasoning. *arXiv preprint arXiv:1908.05739*, 2019.

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

Antoine Bosselut and Yejin Choi. Dynamic knowledge graph construction for zero-shot commonsense question answering. *arXiv preprint arXiv:1911.03876*, 2019.

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

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

Kshitij Fadnis, Kartik Talamadupula, Pavan Kapanipathi, Haque Ishfaq, Salim Roukos, and Achille Fokoue. Heuristics for interpretable knowledge graph contextualization. *arXiv preprint arXiv:1911.02085*, 2019.

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

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

Pavan Kapanipathi, Veronika Thost, Siva Sankalp Patel, Spencer Whitehead, Ibrahim Abdelaziz, Avinash Balakrishnan, Maria Chang, Kshitij Fadnis, Chulaka Gunasekara, Bassem Makni, et al. Infusing knowledge into the textual entailment task using graph convolutional networks. *arXiv preprint arXiv:1911.02060*, 2019.

Souvik Kundu, Tushar Khot, Ashish Sabharwal, and Peter Clark. Exploiting explicit paths for multi-hop reading comprehension. *arXiv preprint arXiv:1811.01127*, 2018.

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

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

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.

Todor Mihaylov and Anette Frank. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. *arXiv preprint arXiv:1805.07858*, 2018.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Timothy Niven and Hung-Yu Kao. Probing neural network comprehension of natural language arguments. *arXiv preprint arXiv:1907.07355*, 2019.

Debjit Paul and Anette Frank. Ranking and selecting multi-hop knowledge paths to better predict human needs. *arXiv preprint arXiv:1904.00676*, 2019.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

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

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

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

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

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

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

Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, et al. Improving natural language inference using external knowledge in the science questions domain.

In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7208–7215, 2019.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302, 2018.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326*, 2018.

Appendix A. Algorithm for Paths Sampling

Algorithm 1 Paths Sampling

```
Input: $\mathcal{G} = (\mathcal{E}, \mathcal{R})$ and a set of all the question entities $\{e^q\}$
Output: A set of triplet paths $\{p\}$.
 1: repeat
 2:   if Do Local Sampling then
 3:     current_node $u \leftarrow uniform\_sample(\mathcal{E})$
 4:   else
 5:     current_node $u \leftarrow uniform\_sample(\{e^q\})$
 6:   end if
 7:   $p \leftarrow \{u\}$
 8:   for $t = 1$ to $T$ do
 9:     $N \leftarrow Neighbor(u)$
10:     next_node $v \leftarrow uniform\_sample(N)$
11:     $M \leftarrow All\_Relations(u, v)$
12:     while TRUE do
13:       $r \leftarrow uniform\_sample(M)$
14:       if $r$ not in $p$ then
15:         BREAK
16:       end if
17:     end while
18:     $p \leftarrow p \cup \{r, v\}$
19:     $u \leftarrow v$
20:   end for
21: until Maximum number of paths achieved.
```

Appendix B. Statistics of Task Datasets and Sampled Paths

Table 5: Statistics of Task Datasets.

Dataset	#Train	#Dev	#Test
CommonsenseQA	8,767	974	1,221
OpenBookQA	4,957	500	500

Table 6: Statistics of Sampled Paths.

Setting	#Paths
Global	538,766
CommonsenseQA	133,612
OpenBookQA	105,155

Appendix C. Discarded Relations

When sampling knowledge paths, we discard some relation types which are regarded to be uninformative and offer little help for answering the questions. They include *relatedto*, *synonym*, *antonym*, *derivedfrom*, *formof*, *etymologicallyderivedfrom* and *etymologicallyrelatedto*.

Appendix D. 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 an 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 the development set no longer decreases for 2 epochs.

For training on task datasets, we search the optimal hyper-parameters based on the classification accuracy on the development set. The initial learning rate is choosing from $\{5e-6, 1e-5, 5e-5\}$. The batch size is chosen from $\{8, 16, 32, 64, 128\}$. A large batch size is achieved by accumulating gradient through several small batches. We also train our model with a warm-up period of 1000 mini-batches and linearly decrease the learning rate. The training lasts until the accuracy on the development set no longer increases for 2 epochs.

Appendix E. Complete Paths from Our Full Method

Q1: Where would you find magazines along side many other printed works?
A: doctor. B^* : *bookstore.* C: market. D: train station. E: mortuary.
Path 1: {magazine, isa, book, atlocation, bookstore}
Path 2: {along side, hascontext, printing, _usedfor, sheet of paper, atlocation, bookstore}
Path 3: {many, _hasproperty, books, atlocation, bookstore}
Path 4: {print, _usedfor, printer, atlocation, bookstore}

Q2: If you want harmony, what is something you should try to do with the world?
A: take time. B. make noise. C. make war. D^**.make peace.* E. make haste.
Path 1: {harmony, _hassubevent, make better world, hasprerequisite, make peace}
Path 2: {try, _hasprerequisite, make peace}
Path 3: {world, _atlocation, human, capableof, make peace}

Q3: Janet was watching the film because she liked what?
A: rejection. B: laughter. C^* : *being entertained.* D: fear. E: bordem.
Path 1 : {film, usedfor, being entertained}
Path 2: {watch, _hasprerequisite, seeing story, causes, being entertained}
Path 3: {janet, isa, person, desires, being entertained}