Dynamic Relevance Graph Network for Knowledge-Aware Question Answering

Anonymous ACL submission

Abstract

This work investigates the challenge of learning and reasoning for Commonsense Question Answering given an external source of knowledge in the form of a knowledge graph. We propose a novel graph neural network architecture, called dynamic relevance graph network (DRGN). DRGN operates on a given KG subgraph based on the question and answers entities and uses the relevance between the nodes to establish new edges dynamically for learning node representations in the graph network. Using the relevance between the graph nodes in learning representations helps the model to not only exploit the existing relationships in the KG subgraph but also recover the missing edges. Moreover, our model improves handling the negative questions due to considering the relevance between the global question node and the graph entities. Our proposed approach shows competitive performance on two QA datasets with commonsense knowledge, CommonsenseQA and OpenbookQA, and improves the state-of-the-art published results.

1 Introduction

Solving Question Answering (QA) problems usually requires both language understanding and human commonsense knowledge. Large-scale pretrained language models (LMs) have achieved success in many QA benchmarks (Rajpurkar et al., 2016, 2018; Min et al., 2019; Yang et al., 2018). However, LMs have difficulties predicting the answer when reasoning over external knowledge is required (Yasunaga et al., 2021; Feng et al., 2020).

Therefore, using the external sources of knowledge in the form of knowledge graphs (KGs) is a common practice in question answering models (Lin et al., 2019; Feng et al., 2020). As shown in Figure 1 taken from the CommonsenseQA benchmark, answering the question requires commonsense reasoning over question and candidate answers, while the external KG provides the required background information. The current state-of-the-art models on CommonsenseQA (Yasunaga et al., 2021) reason over the question and KG by adding the question node to a selected KG entity subgraph and jointly updating the representations by a graph neural model.

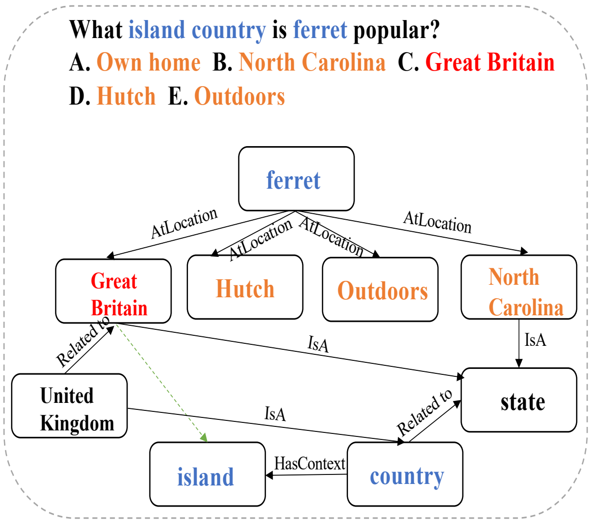

Figure 1: An example of CommonsenseQA benchmark. Given the question (blue boxes) and answer choice (red box and orange boxes), we predict the answer by reasoning over the question information and the extracted KG subgraph.

However, two issues exist in the current approaches: a) the extracted KG subgraph misses some links between entities, which breaks the chain of reasoning or the paths between entities are too long that the current models can not exploit the connections, b) reasoning when negative terms exist in the question, such as no and not, is problematic.

The first above-mentioned issue can be caused by the following reasons. First, the knowledge graph is originally imperfect and does not include those relational edges. Second, when constructing the subgraph, to reduce the graph size, most of the models select the entities that appear in two-hop paths (Feng et al., 2020). Therefore, some intermediate concept nodes and links are missed in the extracted KG subgraph. In such cases, the subgraph cannot establish a complete chain of reasoning. Third, the current models often cannot reason over long paths when there is no direct connection between the involved concepts. Looking back at Figure 1, the KG subgraph misses the direct connection between "Great Britain" and "island" (green arrow), where the term "island" is the most crucial term in the question. For the issue of the negative questions, as Lin et al. points out, KagNet and MHGRN models are not sensitive to the negation words and consequently predict opposite answers. QA-GNN (Yasunaga et al., 2021) model is the first related work to deal with the negative questions. QA-GNN improves the negative reasoning, to some extent, by adding the question node to the graph. However, the challenge still exists.

To solve the above challenges, we propose a novel architecture called Dynamic relevance graph network (DRGN). The motivation of our DRGN is to recover the missing links or establish the shortcut connection to facilitate multi-hop reasoning. In particular, our DRGN model uses a graph network module in which it recovers the missing links or establish a shortcut connection based on the relevancy of the node representations in the KG during the training. The module can potentially capture the connections between distant nodes while benefiting from the existing KG edges. At each convolutional layer of the graph neural network, we compute the similarity of the nodes based on their current representations and build the neighborhoods based on this similarity, forming a relevance matrix accordingly. In this way, the neighborhoods are constructed dynamically. This can be seen as a way to learn new edges based on the relevance of the nodes as the training goes forward in each layer. Moreover, since the graph includes the question node, the relevance between the question node and entity node is computed at every layer, making the model consider the negation information when learning node representations during the training.

The contributions of this work are as follows:
1) Our proposed DRGN architecture exploits the edges in the KG subgraph and uses the relevance between the nodes to establish shortcut connections or recover the missing edges dynamically. This helps capture the full reasoning path in the graph for answering the question.

2) Our model exploits the relevance between question information and the graph entities, which helps in boosting the performance of reasoning over negative questions.

3) Our proposed model obtains competitive results on both CommonsenseQA and OpenbookQA benchmarks. Our analysis demonstrates the significance and effectiveness of the DRGN model.

2 Related Work

2.1 QA using Knowledge Graph

Many recent research has studied methods to augment QA systems with external knowledge. Most of the existing works study pre-trained language models that potentially serve as implicit knowledge bases. To provide more interpretable knowledge, some recent works utilize KGs to the QA models (Feng et al., 2020). However, given that the KGs are usually large and contain many irrelevant nodes to the question in hand, the QA models can not use the KG's information effectively (Feng et al., 2020). Moreover, with larger KGs, the computational complexity of learning over them will increase. Several works deal with this issue by pruning KG nodes based on some metrics (Defferrard et al., 2016; Zhou et al., 2020; Velickovic et al., 2018; Hamilton et al., 2017; Ying et al., 2018).

Furthermore, some recent works use the text context as an additional node in the KG subgraph. For example, Koncel-Kedziorski et al. and Yasunaga et al. introduce sentence node into the graph, while Fang et al. and Zheng and Kordjamshidi add the paragraph node and sentence node to construct a hierarchical graph structure. In our work, we use the question node as the external node and add it to the KG subgraph. Therefore, the graph representations can learn more contextual information by computing the relevance between the question node and graph entity nodes.

2.2 Graph Neural Networks

Graph convolutional network (GCN) (Kipf and Welling, 2017) is a classic multi-layer graph neural network. The node representations in the graph are strongly related to their neighborhood nodes and edges. For each layer of GCN, the node representations capture the information of their neighborhood nodes and edges via message passing and graph convolutional operation. R-GCN is a variation of GCN that deals with the multi-relational graph structure (Schlichtkrull et al., 2018). Li et al. proposes an Adaptive Graph Convolution Network

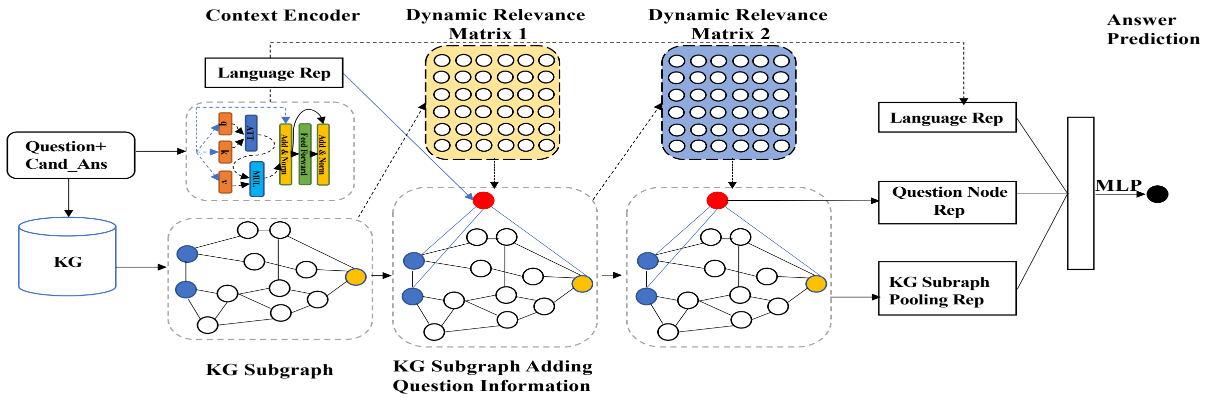

Figure 2: Our proposed DRGN model is composed of the Language Context Encoder module, KG Subgraph Construction module, Graph Neural Network module, and Answer Prediction module. The blue color entity nodes represent the entity existing in the question. The yellow color node represents the answer node. The red color node is the question node. We use different colors to draw the dynamic relevance matrix 1 and 2 because the relevance matrix changes dynamically in each graph neural layer.

(AGCN) to learn the underlying relations and learn the residual graph Laplacian to improve spectral graph performance. Meanwhile, some varients of GCN try to replace the graph Fourier transform. Graph wavelet neural network (GWNN) (Xu et al., 2019) applies the graph wavelet transform to the graph, and achieves better performance compared to the graph Fourier transform in some tasks.

Meanwhile, several models try to use the attention operator on the graphs. For example, the graph attention network (GAT) (Vaswani et al., 2017) uses the self-attention method and multi-head attention strategy to learn the node representations that consider the neighbors of the nodes. Besides, the gated attention network (GaAN) (Zhang et al., 2018) uses self-attention to aggregate the different heads' information. GaAN utilizes the gate mechanism to replace the average operation that is commonly used in the GAT model.

Dynamic GCN (Ye et al., 2020) is another branch of the GCN family. The dynamic graphs are constructed for different input samples. Moreover, Dynamic GCN learns the dynamic graph structure by a context-encoding network, which takes the whole feature map from the convolution neural network as input and directly predicts the adjacency matrix. Besides, Malik et al. proposes Dynamic Graph Convolutional Networks using the Tensor M-Product. Unlike these works, our DRGN model maintains the graph structure statically, but computes the relevance edges dynamically and learns node representation. Besides, our approach uses the existing relationships in the KG, captures the missing ones or finds the shortcut connections by computing the relevance between nodes dynamically. We consider this as learning new edges based on the relevance of the nodes as the training goes forward in each layer.

3 Dynamic Relevance Graph Network

3.1 Problem Formulation

The problem of QA over commonsense knowledge graphs is to choose a correct answer a_{ans} from a set of N candidate answers $\{a_1, a_2, ..., a_n\}$ given input question q and an external knowledge graph (KG). In fact, not the whole KG is input to the problem but a subgraph, $G_{sub} = (V, E)$, is selected inspired by Feng et al. and Yasunaga et al.. The node set V represents entities in the knowledge subgraph, and the edge set E represents the edges between entities.

3.2 Model Description

Figure 2 shows the proposed Dynamic Relevance Graph Network (DRGN) architecture. Our DRGN includes four modules: Language Context Encoder module, KG Subgraph Construction module, Graph Neural Network module, and Answer Prediction module. This section introduces our approach in detail and then explains how to train it with an efficient algorithm.

3.3 Language Context Encoder

For every question q and candidate answer a_i pair, we concatenate them to form Q:

$$Q = [[CLS]; q; [SEP]; a_i], \qquad (1)$$

where [CLS] and [SEP] are the special tokens used by large-scale pre-trained Language Models (LMs). We feed input Q to a pre-trained RoBERTa encoder to obtain the list of token representations $h_Q \in \mathcal{R}^{(|q|+|a|+2)*d}$, where $|q|$ represents the length of the question, while $|a|$ represents the length of the candidate answer. Then we use the [CLS] representation, denoted as, $h_{[cls]} \in \mathcal{R}^d$, as the representation of the question and candidate answer pair.

3.4 KG Subgraph Construction

We use ConceptNet, a general-domain knowledge graph, as the commonsense KG. ConceptNet graph has multiple semantic relational edges, e.g., HasProperty, IsA, AtLocation, etc. We follow Feng et al. work to extract the subgraphs from KG for each example. The approach is to extract a subgraph from KG that contains the entities mentioned in the question and answer choices. The entities are selected with the exact match between n-gram tokens and ConceptNet concepts using some normalization rules. Then another set of entities is added to the subgraph by following the KG paths of two hops of reasoning based on the current entities in the subgraph.

Furthermore, we add the question as a separate node to the subgraph. This node provides an additional question information to the KG subgraph, G_{sub}, as suggested by Yasunaga et al.. We link the question node to entity nodes mentioned in the question. The question node is initialized by the [CLS] representation described in Section 3.3. The initial representation of the other entities is derived from applying RoBERTa and pooling over their contained tokens (Feng et al., 2020).

3.5 Graph Neural Network Module

The basis of our learning representation is Multi-relational Graph Convolutional Network (R-GCN) (Schlichtkrull et al., 2018). R-GCN is an extension of GCN that operates on a graph with multi-relational edges between nodes. In our case, the relation types between entities are taken from the 17 semantic relations from ConceptNet. Meanwhile, an additional type is added to represent the relations between the question node and the entities, which the graph structure is different from previous works. We denote the set of relations as R. $\mathcal{N}_i^r$ denotes the neighbor nodes of node i under relation r, where $r \in R$.

We use $h^{(l)}$ to represent node representations in the l-th layer. In R-GCN, node representations are computed as follows:

$$h_i^{(l+1)} = \sigma\left(W_0^{(l)} h_i^{(l)} + \sum_{r\in R}\sum_{j\in\mathcal{N}_i^r} \frac{1}{d_{i,r}} W_r^{(l)} h_j^{(l)}\right) \in \mathcal{R}^d, \tag{2}$$

$$h^{(l+1)} = [h_0^{(l+1)}; h_1^{(l+1)}; \cdots; h_{|V|}^{(l+1)}] \in \mathcal{R}^{|V|*d}, \tag{3}$$

where $d_{i,r}$ is the normalization factor (Schlichtkrull et al., 2018). $W_r^{(l)}$ is the learned relational weight, and W_0 is the learned self-loop weight, and $|V|$ is the number of nodes in the subgraph.

Our dynamic relevance graph network (DRGN) architecture is the variation of the R-GCN model. To consider the direct relevance between the nodes in learning node representations, we compute the similarity between the nodes in the subgraph dynamically based on their current representations. Then we build the neighborhoods based on this relevance and form a dynamic relevance matrix, M_{rel}, accordingly. This can be seen as a way to learn new edges based on the relevance of the nodes as the training goes forward in each layer. We use inner product to compute the relevance matrix as follows:

$$M_{rel}^{(l+1)} = h^{(l+1)\top} h^{(l+1)} \in \mathcal{R}^{(|V|+1)\cdot(|V|+1)}, \tag{4}$$

where $|V|$ is the graph entity nodes sizes, and the other 1 represents the question node. The relevance matrix re-scales the way that the neighborhood nodes' representations are aggregated in the R-GCN base model. M_{rel} is computed dynamically, and the neighborhood scores change while the representations are further trained. This makes the neighborhood computations dynamic. The forward-pass message passing updates of the nodes denoted by v_i in our proposed relational graph are calculated as follows:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{j\in\mathcal{N}_i^r} \frac{1}{d_{i,r}} W_r^{(l)} \cdot h_j^{(l)} + W_0^{(l)} \cdot h_i^{(l)}\right), \tag{5}$$

$$h_q^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}^q} W_q^{(l)} \cdot F_c([h_q^{(l)}; h_j^{(l)}]) + W_0^{(l)} \cdot h_q^{(l)}\right), \tag{6}$$

$$h'^{(l+1)} = [h_0^{(l+1)}; h_1^{(l+1)}; \cdots; h_{|V|}^{(l+1)}; h_q^{(l+1)}] \in \mathcal{R}^{(|V|+1)*d}, \tag{7}$$

$$h^{(l+1)} = \sigma\left(M_{rel}^{(l+1)} \cdot h'^{(l+1)} \cdot W_g\right) \in \mathcal{R}^{(|V|+1)*d}, \quad (8)$$

where σ is the activation function, W_g is the learning weight, F_c is a two-layer MLP, h_q is the question node representation. We then use the dynamic dynamic relevance matrix computed from the $(l+1)$ graph layer to multiply the node representation matrix $h'^{(l+1)}$ that helps the node representation to learn the direct relevance between the nodes during the massage passing, which is shown in Formula 8.

3.6 Answer Prediction

Given a question q and an answer choice, we use the information from both the language representation $h_{[cls]}$, question node representation h_q learning from the KG subgraph, and the representation pooled from the KG subgraph, G_{sub}, to calculate the scores of the answers. The probability is computed as follow:

$$p(a|Q, G_{sub}) = f_{out}([h_{[cls]}; h_q; pool(h_{G_{sub}})]),$$

where f_{out} is a two-layer MLP. Finally, we choose the highest scored answer from N candidate answers as the prediction output. We use the cross entropy loss to optimize the End-to-End model.

4 Experiments and Results

4.1 Datasets

We evaluate our model on two different QA benchmarks, CommonsenseQA and OpenbookQA. Both benchmarks come with an external knowledge graph. We apply ConceptNet to the external knowledge graph on these two benchmarks.

CommonsenseQA is a QA dataset that requires human commonsense reasoning capacity to answer the questions. Each question in CommonsenseQA has five candidate answers without any extra information. The dataset consists of 12,102 questions.

OpenbookQA is a multiple-choice QA dataset that requires reasoning with commonsense knowledge. The OpenbookQA benchmark is a well-defined subset of science QA (Clark et al., 2018), requiring to find the chain from the open book and commonly known supporting facts. The dataset contains about 6,000 questions.

4.2 Implementation Details

We implemented our DRGN architecture using PyTorch [1]. We use the pre-trained RoBERTa-large (Liu et al., 2019) to encode the question. We use cross-entropy loss and RAdam optimizer (Liu et al., 2020) to train our End-to-End architecture. Moreover, we set the batch size to 16, the maximum text input sequence length to 128. Our model use early stopping strategy during the training. Furthermore, we use 3-layer graph neural module in our experiments. Section 5.3 describes the effect of different number of layers. The learning rate for the LMs is $1e-5$, while the learning rate for the graph module is $1e-3$.

[1] Our code will be available after the paper is accepted.

Models	Dev ACC%	Test ACC%
RoBERTa-no KG	69.68%	67.81%
R-GCN	72.69%	68.41%
GconAttn	72.61%	68.59%
KagNet	73.35%	69.22%
RN	73.65%	69.59%
MHGRN	74.45%	71.11%
QA-GNN	76.54%	73.41%
DRGN	**78.10%**	**75.26%**

Table 1: Dev accuracy and Test accuracy of various models on the CommonsenseQA benchmark.

4.3 Baseline Description

KagNET (Lin et al., 2019) is a path-based model that models the multi-hop relations by extracting relational paths from Knowledge Graph and then encoding paths with an LSTM sequence model.

MHGRN (Feng et al., 2020): Multi-hop Graph Relation Network (MHGRN) is a strong baseline. MHGRN model applies LMs to the question and answer context encoder, uses GNN encoder to learn graph representations, and chooses the candidate answer by these two encoders.

QA-GNN (Yasunaga et al., 2021) is the most recent SOTA model that uses a working graph to train language and KG subgraph. The model jointly reasons over the question and KG and jointly updates the representations. QA-GNN uses GAT as the backbone to do message passing on the graph. To learn the semantic edge information, QA-GNN directly adds the edge feature to the local node feature and cannot learn the global structure of the edges, which is inefficient. However, our model uses the global multi-relational adjacency matrices to learn the edge information. Our model uses the same subgraphs as QA-GNN.

4.4 Result Comparison

Table 1 shows the performance of different models on the CommonsenseQA benchmark. KagNet and MHGRN are two strong baselines. Our model

Models	Dev ACC%	Test ACC%
RoBERTa-large	66.7%	64.8%
R-GCN	65.0%	62.4%
GconAttn	64.5%	61.9%
RN	66.8%	65.2%
MHGRN	68.1 %	66.8%
QA-GNN	68.6 %	67.8%
DRGN	**69.4%**	**69.6%**
AristoRoBERTaV7	79.2%	**77.8%**
UnifiedQA(T5-11B)	-	87.2%
AristoRoBERTaV7+MHGRN	78.6%	80.6%
AristoRoBERTaV7+QA-GNN	-	82.8%
AristoRoBERTaV7+DRGN	**80.2%**	**84.6%**

Table 2: Dev accuracy and Test accuracy of various model performance on the OpenbookQA benchmark.

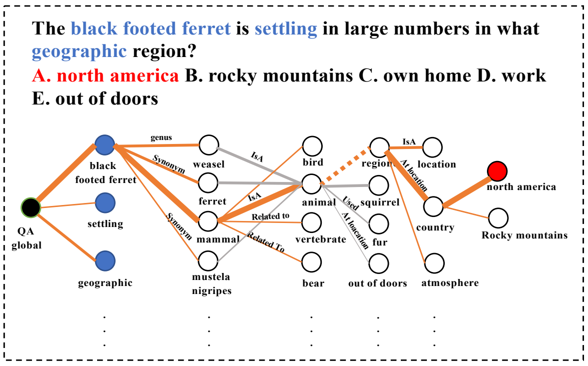

Figure 3: A complete reasoning chain from the question entity node to the candidate answer node. The blue nodes are question entity nodes, and the red node is the gold answer node. The darker orange edges indicate the higher relevance score to the neighborhood node, while the lighter orange edges indicate the lower score. The grey edges are selected from the baseline model.

outperforms the KagNet by 6.04% and MHGRN by 4.15% on CommonsenseQA. Moreover, we obtain SOTA results that shows the effectiveness of our DRGN architecture. Table 2 shows the performance on OpenbookQA benchmark. There are a few recent papers that exploit larger LMs, such as T5 (Raffel et al., 2020) that contains 3 billion parameters (10x larger than our model,) and UnifiedQA (Khashabi et al., 2020) (32x larger). For a fair comparison, we use the same Roberta setting for the input representation when we evaluate OpenbookQA. Our model performance, potentially, will be improved after using these larger LMs. To demonstrate this point, we also use AristoRoBERTaV7 (Clark et al., 2019) as a backbone to train our model. The results show that our model achieves the best performance when using the same larger LMs compared to other strong baseline models. In the same AristoRoBERTaV7 setting, DRGN model has 4% accuracy improvement compared to the MHGRN model and 2% improvement compared to QA-GNN on the OpenbookQA benchmark.

5 Analysis

5.1 Effects of Finding the Line of Reasoning

In this subsection, we analyze the effectiveness of our DRGN that helps in recovering the missing links or establishing a shortcut connection based on the relevancy of the node representations in the KG. As we described in Section 3.4, to keep the graph size small, most of the models construct the KG subgraph via selecting the entities that appear in two-hop paths. Therefore, some intermediate concept nodes and links are missed in the extracted KG subgraph, and the complete reasoning chain from the question entity node to the candidate answer node can not be found.

For example, as shown in Figure 3, the question is “The black footed ferret is settling in large numbers in what geographic region?” and the answer is “North America”. The long reasoning chain includes 5 hops, that is, “black footed ferret → mammal → animal → region → country → North America”. Since the constructed graph misses the link between “animal” and “region”, MHGRN and QA-GNN cannot obtain the complete chain and predict the wrong answer “out of doors” by the grey edges described in the Figure. However, the DRGN model makes a correct prediction. The reason is that we compute the similarity of the nodes based on their learned representations and form a relevance edge accordingly. As we describe in Section 3.4, our model initializes the entity node representation by RoBERTa. The implicit representations of LMs are learned from the huge corpora, and the knowledge is implicitly learned. Therefore, these two words, animal and region, start with an implicit connection. By looking at the relevance matrix, after several layers of graph encoding, we found that the relevance score between “animal” and “region” becomes stronger. This is the primary reason why our DRGN model obtains the complete reasoning chain.

Looking into another example, the question is “Bob spent all of his time analyzing the works of masters because he wanted to become what?” and the answer is “intelligent”. The reasoning chain is “analyze → work → master → learn → intelligent”. Due to the missing link between “master → learn” in KG, the baseline model, MHGRN, can-

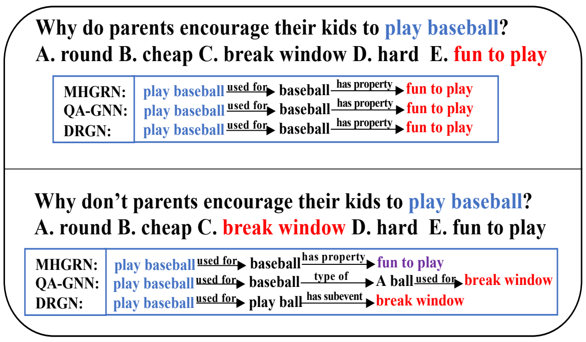
Why do parents encourage their kids to play baseball?
A. round B. cheap C. break window D. hard E. fun to play
MHGRN: play baseball used for baseball has property fun to play
QA-GNN: play baseball used for baseball has property fun to play
DRGN: play baseball used for baseball has property fun to play
Why don't parents encourage their kids to play baseball?
A. round B. cheap C. break window D. hard E. fun to play
MHGRN: play baseball used for baseball has property fun to play
QA-GNN: play baseball used for baseball type of A ball used for break window
DRGN: play baseball used for play ball has subevent break window

Figure 4: The case study of the negation examples. The question in the bottom box includes the negative words. The red color text represents the gold answer, and the purple color represents the wrong answer. In the blue box, each line represents a commonsense reasoning chain from a model.

not obtain the complete reasoning chain. However, our DRGN captures the strong relevance between "master" and "learn" and finds out the complete reasoning chain to predict the correct answer.

5.2 Effects of Handling Negative Questions

While the graph has a broad coverage of knowledge, it can not perform well on handling negation (Yasunaga et al., 2021). Since our dynamic relevance matrix includes the question information, the relevance between the question and graph entities is computed at every graph neural layer while considering the negation in the node representations. Intuitively, this should improve handling the negative question in our model.

To analyze this hypothesis for DRGN architecture, we randomly choose 50 samples from the CommonsenseQA development data. If the question is positive, we add the negation words to generate a negative counterpart question. If the question includes the negation words, we remove them to generate a positive counterpart question. We annotate the answer to the new questions manually. This result is 50 positive samples and their additional 50 negative counterparts of the exact questions (total 100 samples). We compare the performance of various models on questions containing negative words (e.g., no, not, nothing, unlikely). The result is shown in Table 3. We observe that baseline models, KagNet and MHGRN, provide limited improvements over RoBERTa on questions with negation words (+0.2%). However, our DRGN model exhibits a huge boost (+7.8%). Moreover, the DRGN model has a 3.5% accuracy improvement compared to the QA-GNN model, demonstrating the effectiveness of handling negative questions that experimentally confirm our hypothesis. An additional ablation study in Table 5 confirms this idea further. When removing the question information from DRGN, we observe that the performance on negation becomes close to the MHGRN.

Models	Dev ACC % w/o negative	Dev ACC% negative
RoBERTa-large	69.1%	52.7%
MHGRN	72.5 %	52.9%
QA-GNN	74.3 %	58.0%
DRGN	**76.6%**	**61.5%**

Table 3: Accuracy of the positive questions and negative questions on 100 samples from CommonsenseQA.

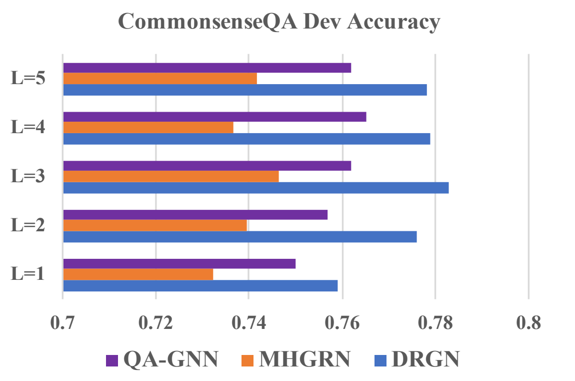

Figure 5: The Effect of different number of layers in QA-GNN, MHGRN, and DRGN models.

Figure 4 shows case studies about the commonsense question and negative question. As is shown in the figure, all the models obtain the same reasoning chain "play baseball-(used for)→ baseball-(has property)→ fun to play", including MHGRN, QAGNN, and our architecture. However, when adding the negative words, MHGRN obtains the same reasoning chain as the positive situation, while QAGNN and our DRGN find the correct reasoning chain. One interesting finding is that our DRGN finds out the shortcut connection and uses fewer hops to establish the reasoning chain, which is shown in Figure 4. The chain of reasoning is found by looking at the neighborhood matrices in the model. For DRGN, this neighborhood is the outcome of multiplication by relevance matrix.

5.3 Effects of Number of Graph Layers

We evaluate the effects of multiple layers l for the baseline models and our DRGN by evaluating its performance on the CommonsenseQA development dataset. As shown in Figure 5, the increase

Models	Time	Space
l-layer KagNet	$O(\|R\|^l \|V\|^{l+1} l)$	$O(\|R\|^l \|V\|^{l+1} l)$
l-layer MHGRN	$O(\|R\|^2 \|V\|^2 l)$	$O(\|R\| \|V\| l)$
l-layer QA-GNN	$O(\|V\|^2 l)$	$O(\|R\| \|V\| l)$
l-layer DRGN	$O(\|R\|^2 \|V\|^2 l)$	$O(\|R\| \|V\|^2 l)$

Table 4: The time complexity and space complexity comparison between DRGN and baseline models.

Models	Dev ACC
DRGN w/o KG subgraph	69.6%
w/o relational edges in graph	71.3%
w/o question node in graph	74.7%
DRGN	**78.1%**

Table 5: Ablation Study on CommonsenseQA dataset.

of l continues to bring benefits until $l = 3$ for our DRGN. However, performance begins to drop when $l > 3$. This might be because of the noise in the long reasoning chains in the KG. We compare the performance after adding each layer for MHGRN, QA-GNN, and our DRGN. We observe that DRGN consistently achieves the best performance with the same number of layers as the baselines.

Table 4 shows the time complexity and the space complexity comparison between our DRGN model and baseline model. We compare the computational complexity based on the number of layers l, the number of nodes V, and the number of relations R. We report this for the baseline models based on their papers (Yasunaga et al., 2021). Our model and MHGRN have the same time complexity because both models use the R-GCN model as the backbone. Besides, QA-GNN utilizes graph attention Transformer (GAT) (Vaswani et al., 2017) to learn the graph node representation. QA-GNN directly adds the edge feature to the local node feature without the global semantic relational adjacency matrices. The scope of the GAT attention is still in the predefined neighborhood area in the KG subgraph. However, in our DRGN model, we consider all nodes in the subgraph. It is hard to evaluate which of the two models, R-GCN and GAT, is more effective. However, after adding the dynamic relevance information, our R-GCN based DRGN model achieves better performance compared to GAT based QA-GNN baseline.

For the space complexity, our model's space complexity is larger than MHGRN because DRGN introduces the extra dynamic relevance matrix. However, this cost depends on the size of the subgraph which is usually small and meanwhile it leads to 5% performance improvement.

5.4 Ablation Study

To evaluate the effectiveness of various components of our DRGN, we perform an ablation study on the CommonsenseQA development dataset. Table 5 shows the results of ablation study. First, we removed the whole commonsense subgraph. We observe that our model without the subgraph obtains 69.6% on the CommonsenseQA dataset. This shows how the language model can answer the question without the external KG, which is not high-performing but still impressive. Second, we kept the KG subgraph but removed all the relational edge information from the subgraph (described in section 3.5). Without the relational edges, the accuracy is 71.3% on the CommonsenseQA benchmark with a 6.8% gap compared to the DRGN model. This result shows that the multiple relational edges helps the DRGN learn better representations and obtain higher performance. Third, we keep the subgraph and remove the question node. In other words, we lose an additional type of relationship between the question node and the graph entities. More importantly, we cannot obtain the relevance between question information and entity node in the dynamic relevance matrix. The accuracy of the DRGN decreases from 78.1% to 74.7%. It demonstrates the importance of the relevance between the question information and the KG subgraph.

6 Conclusion

In this paper, we propose a novel dynamic relevance graph neural network (DRGN) model to learn commonsense question answering given an external source of knowledge in the form of a KG. Our model learns the graph node representation that exploits the existing relations in KG and recovers the missing connections or establishes the shortcut connections based on measuring the relevance scores of the nodes dynamically during training. Our quantitative and qualitative analysis shows that the proposed approach facilitates finding the chain of reasoning for answering the questions that need multiple hops of reasoning. Furthermore, since DRGN considers the relevance between the question and graph entities, our model improves the performance on the negative questions. Our proposed approach shows SOTA performance on two QA benchmarks, including CommonsenseQA and OpenbookQA.

References

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Peter Clark, Oren Etzioni, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, Niket Tandon, et al. 2019. From'f'to'a'on the ny regents science exams: An overview of the aristo project. *arXiv preprint arXiv:1909.01958*.

M. Defferrard, X. Bresson, and P. Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*.

Yuwei Fang, S. Sun, Zhe Gan, Rohit Radhakrishna Pillai, Shuohang Wang, and Jingjing Liu. 2020. Hierarchical graph network for multi-hop question answering. In *EMNLP*.

Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable multi-hop relational reasoning for knowledge-aware question answering. In *EMNLP*.

William L. Hamilton, Zhitao Ying, and J. Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*.

D. Khashabi, S. Min, T. Khot, A. Sabhwaral, O. Tafjord, P. Clark, and H. Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single qa system. *EMNLP - findings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *NAACL*.

Ruoyu Li, S. Wang, Feiyun Zhu, and J. Huang. 2018. Adaptive graph convolutional neural networks. In *AAAI*.

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

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the variance of the adaptive learning rate and beyond. In *ICLR*.

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*.

Osman Asif Malik, Shashanka Ubaru, Lior Horesh, Misha E Kilmer, and Haim Avron. 2021. Dynamic graph convolutional networks using the tensor m-product. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 729–737. SIAM.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *ACL*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL*.

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

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.

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.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. 2019. Graph wavelet neural network. In *ICLR*.

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

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *NAACL*.

Fanfan Ye, Shiliang Pu, Qiaoyong Zhong, Chao Li, Di Xie, and Huiming Tang. 2020. Dynamic gcn: Context-enriched topology learning for skeleton-based action recognition. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 55–63.

Rex Ying, Ruining He, K. Chen, Pong Eksombatchai, William L. Hamilton, and J. Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and D. Yeung. 2018. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *UAI*.

Chen Zheng and Parisa Kordjamshidi. 2020. SRLGRN: Semantic role labeling graph reasoning network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8881–8891, Online. Association for Computational Linguistics.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and M. Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.