COMMONSENSEQA Challenge: Improving Question Answering with Knowledge Graphs

Yunguan Fu **Sorawit Saengkyongam** **Yun Seok Kang**

Abstract

In recent years, neural networks have achieved remarkable success on various question answering tasks. However, they still struggle to answer commonsense-based questions which have no passages and thus require information from external sources. Inspired by (Zhong et al., 2018), we propose to extract concept graphs from sentences and use the graph structures to help identify the correct answers. Our proposed method obtained an accuracy of 62.56% (an absolute improvement of 1.05% over the vanilla BERT-based model). We also introduce some handcrafted rules to reformulate questions, which results in a higher accuracy of 63.93%. Our results suggest that including the graph structures extracted from external knowledge base is beneficial for commonsense-based question answering.

1 Introduction

Question Answering (QA) is the task of comprehending text and answering questions in natural language processing (NLP). Generally, there are two different types of tasks: reading comprehensions, and open-domain QA tasks. For the reading comprehension task, most datasets, including the DeepMind CNN dataset (Hermann et al., 2015) and the Stanford question answering dataset (SQuAD) (Rajpurkar et al., 2016), frame the task as answering questions about a relevant passage, which could be a paragraph or a document. On the other hand, for the open-domain QA task, little information is provided, and the answers must be inferred from an external knowledge source.

Recently, there has been an increasing number of studies on answering multiple-choice questions (MCQs) in reading comprehension (Lai et al., 2017; Ostermann et al., 2018; Sun et al., 2019). However, answering MCQs for open-domain QA is a more open problem, as there is no clear way of injecting external knowledge. This is undoubtedly an important research area, as exploiting external knowledge is ubiquitous in human reading and comprehension.

In this paper, we focus on the recently released COMMONSENSEQA v1.0 dataset (Talmor et al., 2018), an open-domain QA task of answering MCQs built by crowd workers. The dataset contains 9,500 MCQs with 3 answer choices each, where exactly one of them is correct. All answers themselves are concepts from CONCEPTNET (Speer et al., 2017) and they share at least one common neighbour belonging to the question. However, as the questions are designed by crowd workers, external human knowledge is also injected into the questions, making them more difficult to answer. As demonstrated in (Talmor et al., 2018), a pre-trained language model obtains only 54.8% accuracy, significantly lower than human performance at 95.3%. As a passing remark, version 1.1 of the COMMONSENSEQA dataset was released very recently, which contains 12,247 questions with 5 answer choices. However, in this paper we only look at version 1.0.

Inspired by (Zhong et al., 2018) and their promising results on the AI2 Reasoning Challenge dataset, we propose a mixed model, which includes a word-based model to capture semantic information, and a graph-based model that extracts graphs from CONCEPTNET. By testing different models, including graph neural networks, we attempt to answer our main research question: *do knowledge graphs help in open-domain QA tasks?*

Regarding the structure of this paper, we describe the related work in Section 2 and provide some background material in Section 3. Then, in Section 4, we give a full exposition of our proposed approach to tackle the COMMONSENSEQA Challenge. Following this, Section 5 describes the setup of experiments. The results are presented and analysed in Section 6. Finally, the conclusion and discussion of possible directions for future work are included in Section 7.

2 Related Work

In this section, we outline some of the recent advances in QA tasks, followed by graph-based approaches in more general NLP problems.

2.1 Language Modelling

Over the last few years, unsupervised pre-trained language modelling (LM) has been shown to be the dominant approach in NLP. (Peters et al., 2018) proposed embeddings from language models (ELMo) using the bidirectional long short-term memory (LSTM) mechanism, and achieved impressive results on various tasks, including question answering, textual entailment, and sentiment analysis. Next, based on Transformer (Vaswani et al., 2017) – a neural network architecture using attention mechanisms – OpenAI introduced the generative pre-training (GPT) model (Radford et al., 2018) that performs left-to-right encoding and decoding simultaneously. And most recently, with the idea of using bidirectional Transformers to jointly condition on both the left and the right context, Google proposed the bidirectional encoder representations from Transformers (BERT) language model (Devlin et al., 2018), which achieved the state-of-the-art (SOTA) result on multiple tasks including the SQuAD dataset. Despite all the above impressive results, these methods that rely solely on unsupervised language modelling still substantially under-perform compared to the human baselines for QA tasks requiring commonsense knowledge (Zellers et al., 2018; Talmor et al., 2018).

2.2 External Knowledge Base

Several recent works have explored using knowledge graphs in open-domain QA tasks. (Lin et al., 2017) extracted commonsense knowledge from heterogeneous knowledge sources by learning correlations between concepts with point-wise mutual information. (Yang and Mitchell, 2019) leveraged concepts by augmenting the dense representations of the networks with symbolic features derived from the knowledge graphs. (Wang et al., 2018a) presented Three-way Attentive Networks (TRiAN) to explicitly model commonsense knowledge by using relation embeddings based on CONCEPTNET as additional features. (Ni et al., 2018) proposed the essential term selector, which identifies the most important words to reformulate more efficient queries and search for related evidence. Finally, most recently, (Zhong et al., 2018) explored an approach to pre-train commonsense-based functions with TriAN in order to capture the semantic relationships between concepts.

2.3 Graph-based Models

Graph structure is central to many NLP tasks, including link predictions in citation networks and entity classification in knowledge bases. To capture the graph structure in representations, DeepWalk (Perozzi et al., 2014) and node2vec (Grover and Leskovec, 2016) are based on random walks on graphs. Also, inspired by doc2vec (Le and Mikolov, 2014), graph2vec (Narayanan et al., 2017) uses the skip-gram network to learn graph representations directly. Alternatively, based on neighbourhoods, gated graph sequence neural networks (GGNNs) (Li et al., 2016) and graph convolutional networks (GCNs) (Kipf and Welling, 2016) were proposed to learn vertex embeddings. Since then, many graph-based neural networks have been proposed, including the relational graph convolutional networks (R-GCNs) (Schlichtkrull et al., 2018) that also takes edge types into consideration. More recently, (Sorokin and Gurevych, 2018) extended GGNNs using TRiAN to learn both vertex and edge representations, which demonstrated great potential of graph-based networks.

3 Background

In this section, we briefly review the main works that appear in our approach.

3.1 BERT

BERT is a pre-trained multi-layer bidirectional Transformer encoder which is trained on two unsupervised prediction tasks: masked LM and next sentence prediction. The masked LM task requires the model to predict randomly masked tokens, so that the token representations would be context-sensitive. In the next sentence prediction task, the model is asked to predict whether two given sentences are adjacent. This helps the model to understand the relationships between sentences, which would not be captured directly by LM.

BERT can easily be applied to supervised downstream tasks by stacking layers on top. To fit the supervised data, we can either fine-tune the model by updating all parameters, or use a feature-based approach to extract features from BERT as fixed inputs. Compared to fine-tuning, the feature-based approach has a lower computation cost because we can pre-compute the representations and quickly test different models, since we do not need to use BERT which has 110 million parameters. However, the fine-tuned model is generally more powerful, as observed in Table 7 of (Devlin et al., 2018).

3.2 R-GCNs

Consider a directed multi-graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ with nodes $v_i \in \mathcal{V}$ and edges $(v_i, r, v_j) \in \mathcal{E}$, where $r \in \mathcal{R}$ is a relation directed from v_i to v_j. R-GCNs aim to learn node embeddings that take both the neighbour nodes and their relations into consideration. More specifically, given a node i, its new embedding h_i^{new} is calculated by

$$h_i^{\text{new}} = \sigma\big(\sum_{r\in\mathcal{R}} \sum_{j\in\mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r h_j + W_0 h_i\big),$$

where N_i^r denotes the indices of neighbours under relation r, $c_{i,r}$ is a normalization term, W_r is a shared linear projection for relation r, and σ is the activation function.

Compared to the GCN, this architecture is able to extract more information from relation types by using different transformations for different relations. Furthermore, to reduce the number of parameters and to prevent overfitting, authors also proposed a basis decomposition of W_r that builds W_r using basis transformations $\{V_b\}$. Instead of having a separate transformation matrix W_r, we use a linear combination of basis transformations to construct W_r, i.e. $W_r = \sum_{b=1}^{B} a_{rb} V_b$, so that the number of parameters would only increase linearly with respect to the number of relation types.

4 Methods

The proposed model consists of two sub-models: a word-based model, to capture the semantic meanings of words and sentences; and a graph-based model, to capture the hidden structure between related concepts. We describe the details of these two sub-models, as well as different approaches to combine them into a final model.

4.1 Word-based Model

4.1.1 BERT

Our word-based model is built based on the pre-trained BERT model. Given a question-answer pair (q, a), we concatenate the answer to the question by ([CLS]+ q +[SEP]+ a +[SEP]) and pass the result to BERT to get the token representations. As suggested in (Devlin et al., 2018), there are multiple choices for contextual token representations. We could either use [CLS] token's final embedding to represent the whole pair, or construct an embedding based on all the tokens. For the latter case, embeddings can be extracted from different layers. After obtaining the embeddings of each token in the question-answer pair, we could obtain the final representation of the pair by mean pooling, or optionally adding a multi-head self-attention layer on top before performing the mean pooling. Finally, the embedding of the pair is passed into a multi-layer perceptron (MLP) to produce a final score.

4.1.2 Question Reformation

When we apply BERT to our QA task, we are simply concatenating the answer to the question, hoping that the model is able to distinguish the relationship between them. However, the answer is at most four words long, and so the concatenated result is usually not a proper sentence. Therefore, in this section, we consider another approach to effectively combine the question and the answer.

Figure 1 illustrates that majority of the questions contains "*What*" or "*Where*". Further, by closely looking at these questions, we found that the structure is often simple – for instance, "*What is likely long on a cat?*". This observation motivated us to reform the questions to integrate the answer candidate into the question. Taking the question above as an example, to incorporate the answer "*Whiskers*", we can simply replace "*What*" with the candidate to obtain "*Whiskers is likely long on a cat.*", which is more natural in terms of language modelling.

Four simple rules were crafted (shown in Table 1), which mainly replace "what" or "where" by the answer candidate. These rules cover around one-third of the questions in training/dev/test set. Among these reformable questions, the first and second rule roughly account for 25% and 60% respectively.

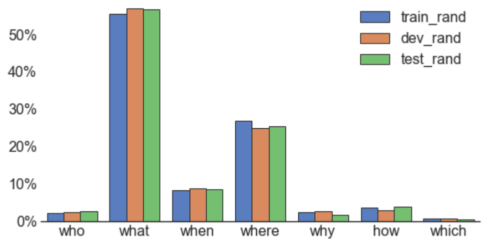

Figure 1: Frequency of the different types of questions.

Original	Reformed
What is/are [...]?	'A' is/are [...].
[...] what?	[...] 'A'.
[...] where?	[...] 'A'.
Where do/does [...]?	[...] at 'A'.

Table 1: Rules of reforming a question. 'A' is the answer choice.

4.2 Graph-based Model

4.2.1 Graph Construction

In order to use the structure between concepts in questions and answers, we need to first extract the concepts and build the graph for each question-answer pair.

Concept Extraction Similar to (Zhong et al., 2018), we extract the $\{1, 2, 3\}$-grams from questions and $\{1, 2, 3, 4\}$-grams from answer candidates. Then, we only keep the n-grams that exist in ConceptNet. The common stopwords are not removed because some stopwords appear as answer candidates.

Further, as described in (Talmor et al., 2018), all answer candidates share a common question concept and all candidates themselves are concepts. Thus, we search the ConceptNet to identify the shared question concepts and call them as *question key concepts*. We note that there may be multiple key concepts and also that the key concepts may not appear in the questions. The other question concepts are referred to as *question context concepts*. Similarly, we refer the concept of the answer candidate as the *answer key concept* and the derived ones as *answer context concepts*.

Graph Construction The graph we construct for each question-answer pair is a bipartite graph between question concepts and answer concepts, i.e. the relations between question concepts or answer concepts are not considered. The intuition is that, since all answer candidates are related to the key concept of the question, if one candidate has more relations to other concepts in the question, it has a greater chance of being the correct answer.

For a question concept and an answer concept, we consider two types of relations: direct and indirect. Direct relation means that the concepts are directly connected in ConceptNet, and indirect relation means that the concepts share a common neighbour not being directly connected. Also, the concepts are considered to be connected regardless of the orientation of the edge. We denote the set of common neighbours as *medium concepts*, since they are the medium of connecting question/answer concepts.

4.2.2 Embedding-based Model

Given a graph for a question-answer pair, the embedding-based model is based on the pairs of question-answer concepts $\{(c_q^i, c_a^i)\}$ which have direct/indirect relations.

Each concept c consists of at most four words $\{w_c^j\}$ and has an initial embedding e_c. The embedding e_c could either be the embedding provided by ConceptNet or the mean of word embeddings $e_{w_c^j}$ where the word embeddings are extracted from BERT. To reduce the size of the networks, the initial embeddings are projected into a lower dimensional space with a linear projection before further processing. Where there is no ambiguity, we will use the same notation for the projected embedding.

For each pair (c_q^i, c_a^i), we concatenate the embeddings of concepts, i.e. $e_{qa}^i = [e_{c_q^i}; e_{c_a^i}]$, and pass it to an MLP which produces a single score $s^i = \text{MLP}(e_{qa}^i)$. The score of the answer candidate is then the sum of these scores, i.e. $s = \sum_i s^i$. The answer with highest score will be chosen as predicted answer.

In addition, before the concatenating embeddings, we can use the R-GCNs to further develop the embeddings of concepts to take into consideration the relations between concepts. We could either apply R-GCNs on the whole graph of all related concepts in dataset, or only on the union of the graphs in a batch.

4.2.3 Feature-based Model

Instead of dealing with all the concept embeddings, we also consider a simpler model in which we obtain a score for each QA pair depending solely on the topological structure of the extracted graph. More precisely, given a graph for a question-answer pair, we extract a series of handcrafted features and pass them to an MLP to produce a score s.

We count the number of concepts in different sets (question key/context concepts, answer key/context concepts, and medium concepts) and the number of edges between these sets (e.g. number of edges between question context concepts and answer key concepts). We also consider the sum of a selection of these features as a new feature (e.g. number of total question concepts). For example, if we only count the number of direct/indirect relations, we have only two features; if we consider all types of relations regardless of their types, we have 33 features; if we distinguish the relation types, we can have up to 1217 features.

4.3 Mixed Model

To combine the word-based model and the graph-based model, the most straightforward way is to add the scores from each model for a given answer candidate.

For the graph feature-based model, we could have a more complex approach. Denote the embedding of the question-answer pair in word-based model as e^w_{qa} and the features of the graph as e^g_{qa}. We first use a linear layer (with ReLU activation) L to develop the graph features and then concatenate them to the word-based embedding, i.e. $e_{qa} = [e^w_{qa}; L(e^g_{qa})]$. This mixed embedding is then forwarded to an MLP to produce a score.

5 Experiments

To investigate our research question "*do knowledge graphs help in open-domain QA tasks?*", we conduct a series of experiments to compare the performance of different embeddings, structures, and sizes of models. Due to limited time and computing resources, we choose to only focus on the COMMONSENSEQA dataset v1.0 with the random split. In addition, we take the feature-based approach to apply BERT, because its training is faster and also because fine-tuning a model almost always improves the performance and our research interest is not about having the best performance on this dataset.

5.1 Metrics

To evaluate the performances of different models on the COMMONSENSEQA dataset, we mainly use the accuracy on the development set as our metric, because the labels for the test set are not available. The reported accuracy is the average of the best dev accuracy over three runs. Each run stops if the highest accuracy on dev set does not increase for five epochs.

5.2 Comparison of Models

As our model contains two sub-models (word-based and graph-based), we first need to test each sub-model separately before testing the combined model. Keeping this in mind, the experiments are designed as the following:

Word-based Model First, we compare the different features extracted from BERT while concatenating the question and answer candidate directly. Then, we apply the reformation rules to a subset of the dataset and analyse whether these handcrafted rules do help.

Graph-based Model Considering the graph embedding based model, we first compare the different concept embeddings. Then, we test if RGCN layers improve the performance by further developing the embeddings. Afterwards, the embedding-based model is compared to the feature-based models.

Mixed Model Finally, we compare the performance of sub-models and the mixed model to answer our main research question of whether graphs are helpful in answering the questions. We compare the different approaches of combining these two sub-models to produce a better mixed model.

5.3 Hyperparameters and Packages

The default parameters are as follows. All neural networks are trained using the Adam optimizer with learning rate 5×10^{-5} and L2 regularisation weight 10^{-4}. The mini-batch size is 64. The hidden size of all layers in MLP and attentions are 64, and every MLP contains only one hidden layer (therefore three layers in total). The dropout rate is set to 0.1 in multi-head attentions. For mixed models adding logits, the weight of graph-based logits is 0.5.

To accelerate the pre-processing step, the distributed framework Ray (Moritz et al., 2018) is used. Also, the framework Deep Graph Library (Wang et al., 2018b) and its implementation were used to implement our R-GCN layers. All the experiments were conducted on a laptop equipped with a GTX 1050 Ti Max-Q GPU, an Intel i7-8750H CPU, and 16 GB RAMs.

6 Results

6.1 Word-based Model

6.1.1 Word Embedding

As mentioned above, we first compared the word-based model with different embeddings from BERT. We tested the embeddings extracted from the first layer, the last layer, the second-to-last layer and the sum of the last four layers. We also tested the influence of the additional multi-head self-attention layer.

The results are presented in Table 2. The best model was to use the embeddings from the second-to-last layer of BERT and to apply a multi-head self-attention layer on top followed by mean pooling. The performance of the first layer's embeddings are poor, because this does not capture high-level semantic information. The last layer's embeddings or the embedding of [CLS] are also not optimal because these are too specific to the original unsupervised tasks in BERT's model. Therefore, the second-to-last layer yields the highest performance by capturing complex semantic information while not being too specific. Lastly, adding a self-attention layer significantly increases the performance as it enables the model to focus on certain key words.

Based on these results, for the subsequent experiments, we decided to always use the embeddings from the second-to-last layer of BERT and to apply the self-attention before the mean pooling aggregation.

Embedding	Model	
	Mean	Mean + Self-Attn
CLS	51.23%	/
first	39.79%	39.79%
last	54.84%	58.00%
sec-to-last	55.26%	**61.51%**
sum last four	55.12%	60.04%

Table 2: Comparison of different word embeddings.

6.1.2 Question Reformation

To test if the reformation of questions helps, we generated a subset of the original dataset to include only the reformable questions. The filtered dataset contains 2603/341/305 questions in train/dev/test set respectively. On this subset, the word-based model achieved 66.18% accuracy with reformation and 58.94% with original questions. This shows that the handcrafted rules help in our task. To better understand the effectiveness of each rule, we calculated the mean accuracy of different models for each rule, with results shown in Figure 2, as the rules are mutually exclusive. We observe that the reformation improves the accuracy significantly on the second and third rule.

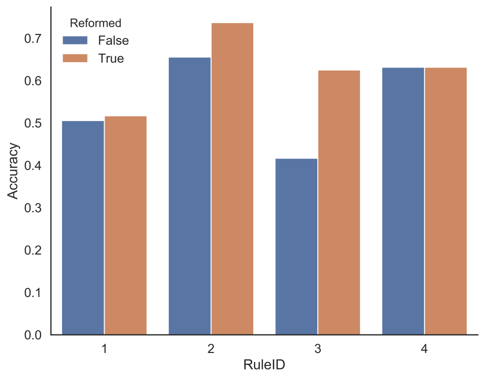

Figure 2: Accuracy on different rules. Blue is without reformation and orange is with reformation.

6.2 Graph-based Model

The graphs are generated as described in Section 4.2.1. Across the 9,500 questions and 28,500 graphs (because there are three question-answer pairs per question), a question has 3.66 ± 2.54 concepts and an answer has 1.80 ± 0.98 concepts. The number of direct relations is 3.17 ± 3.20 and the number of indirect relations is 4.67 ± 4.74. The maximum number of concepts in questions is 34 and for answers is 6. The maximum number of relations are 50 and 59 for direct and indirect relations. These statistics show that the graphs that we generated are relatively small. This fits our expectations, as the questions are often very short (most under 20 words). We tested several other methods to generate graphs, but they sometimes generated graphs containing hundreds of nodes, making the computations too expensive.

6.2.1 Embedding-based Model

As in the word-based model, we have multiple choices for the initial embeddings of concepts. We can either use the embedding provided by CONCEPTNET, or extract embeddings from BERT as before. In addition, we tested the pre-trained word embeddings used in the first layer of BERT that does not depend on the context. For concepts containing multiple words, we used the average of word embeddings as concept embedding. We also tested the random embeddings to check if the embedding is crucial to the performance.

The results are presented in Table 3. Using word embeddings from BERT gave the best performance; but the performance of using random embeddings is also on par. Here, the input embeddings were fixed and not trainable, and during the experiments we noticed that the neural network overfitted the training set quickly (in less than ten epochs). By making the embeddings trainable, we observed a faster overfitting as well as worse performance, e.g. using word embeddings as trainable inputs gave an accuracy of 42.49%. From these results, we decided to fix the embeddings as non-trainable parameters for the experiments that follow.

random	concept	word	CLS
42.49%	43.05%	**43.44%**	42.67%
first	last	sec-to-last	sum last four
42.91%	42.28%	41.19%	41.47%

Table 3: Comparison of different concept embeddings.

As the concept embeddings only capture the semantic meaning without considering the hidden structure between concepts, we extended the model by appending R-GCN layers to inject concept relations into embeddings. Due to the limit of compute resources, we chose to do this on graphs generated by batches, and we also had to reduce the dimension of embeddings of concepts from 64 to 32. We compared the performance without R-GCN layers and with different numbers of layers. The initial embeddings are extracted from BERT's word embedding.

The results are presented in Table 4. We see that the performance is improved by adding R-GCN layers, but the increase is marginal and adding more layers does not seem to give any improvements. Combined with the competitive performance of the random embedding shown above, we believe that, at least for our model on this task, the embeddings of concepts are not crucial.

6.2.2 Feature-based Model

As the random embedding provides a relatively similar performance as pre-trained embeddings, we believe that the capacity of the network does not come from the embeddings. Therefore, we tested the model again by replacing all edges by the same edge which connects a certain concept to itself while keeping the edge mask unchanged. It means that the network has no information on the details of edges but their numbers. Surprisingly, this model also gave a competitive accuracy of 42.18%.

# layers	Acc	# Params
0	43.47%	41.38K
1	**43.86%**	76.51K
2	43.61%	111.65K
3	43.71%	146.79K

Table 4: Comparison of R-GCN layers.

As a result, we designed a new model which does not consider the nodes of edges. The features for a question-answer pair is the number of different types of edges in the graph. We tested three models which have correspondingly 2, 33, and 1217 features for each graph. The results and the number of parameters in each model are given in Table 5. As expected, the features we extracted are able to guide the neural network. Particularly, by using 1217 features, the model achieved a better performance without access to any concept embeddings.

# features	2	33	1217
Acc	42.70%	42.91%	**44.35%**
# Params	4.42K	6.40K	82.18K

Table 5: Comparison of different graph features.

6.3 Mixed Model

To combine the models, we chose the following four basic models and tested different combinations between them: 1) **Word**, a word-based model with embeddings extracted from the second-to-last layer of BERT with a top self-attention layer; 2) **Embed**, a graph embedding based model with a single R-GCN layer; 3) **Feature$_{\textbf{BASE}}$**, a graph feature based model with 33 features; and 4) **Feature$_{\textbf{LARGE}}$**, a graph feature based model with 1217 features. To combine the model, we tested two approaches: i) adding logits of word and graph models, and ii) concatenating their embeddings before computing the final logit. Since it is not straightforward to obtain a fixed length embedding for each QA pair in graph embedding based model, we only considered the concatenation approach for the graph features-based model.

The models' performances and the number of their trainable parameters are presented in Table 6. The best model is to combine the word-based and the graph feature based models by concatenation, which achieves 62.56% accuracy, an absolute improvement of over 1%. Also, we observe that by adding graph features, different approaches all result in some improvement. This proves that adding external information from graphs can help in answering the questions.

Motivated by the improvements shown in Section 6.1.2, we also applied the reformation to several models, whose results are given in Table 7. By adding logits of word-based model and graph feature based model, we achieved a higher accuracy of 63.93%. However, the concatenation provided a relatively bad performance, even worse than without graph features. This may be due to overfitting, but it still remains unclear.

Model	Acc	# Params
Word	61.51%	1.48M
Embed	43.86%	76.51K
$\text{Feature}_{\text{BASE}}$	42.91%	6.40K
$\text{Feature}_{\text{LARGE}}$	44.35%	82.18K
Word+Embed	61.09%	1.56M
$\text{Word+Feature}_{\text{BASE}}$	61.93%	1.49M
$\text{Word+Feature}_{\text{LARGE}}$	61.58%	1.56M
[Word; $\text{Feature}_{\text{BASE}}$]	61.72%	1.58M
[Word; $\text{Feature}_{\text{LARGE}}$]	**62.56%**	2.19M

Table 6: Comparison of different models, where + means the logits are summed up, and [;] means concatenation. $\text{Feature}_{\text{BASE}}$ uses 33 features and $\text{Feature}_{\text{LARGE}}$ uses 1217 features.

Model	Acc	# Params
Word	62.95%	1.48M
$\text{Word+Feature}_{\text{LARGE}}$	**63.93%**	1.56M
[Word; $\text{Feature}_{\text{LARGE}}$]	56.04%	2.19M

Table 7: Comparison of different models with reformation.

7 Conclusion and Future Works

In this paper, we investigated the research question: "*do knowledge graphs help in question answering?*" To answer this question, we proposed a model which extracts concept graphs from question-answer pairs and builds features based on the graph structure. On the COMMONSENSEQA dataset, the proposed model obtained 62.56% accuracy, which is an absolute improvement of 1.05% over the BERT-based baseline. A series of experiments demonstrated that the graph topology is important and helpful for answering these commonsense-based questions. We also tested the graph neural networks R-GCNs; but the increase in performance was disappointing. To consider language modelling, we also investigated some handcrafted rules that reformulated a question-answer pair into a proper sentence. Results showed that these rules have the potential to significantly improve the models' performances.

Although the proposed method improved the BERT-based baseline, the accuracy is still far away from human performance. One possible reason is that the size of graph we generate is too small that it does not contain enough information, and therefore the graph neural network yields limited performance. Future work should validate this method on other datasets such as the AI2 Reasoning Challenge dataset. We could also try to generate larger graphs to capture more information. Moreover, as the version 1.1 of COMMONSENSEQA dataset has been released, which contains more questions and more candidates for each question, it would be a meaningful yet simple extension to test our approach on this newer dataset.

References

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

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

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

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: large-scale reading comprehension dataset from examinations. *CoRR*, abs/1704.04683.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated graph sequence neural networks. *CoRR*, abs/1511.05493.

Hongyu Lin, Le Sun, and Xianpei Han. 2017. Reasoning with heterogeneous knowledge for commonsense machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2032–2043.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. 2018. Ray: A distributed framework for emerging {AI} applications. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 561–577.

Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*.

Jianmo Ni, Chenguang Zhu, Weizhu Chen, and Julian McAuley. 2018. Learning to attend on essential terms: An enhanced retriever-reader model for scientific question answering. *arXiv preprint arXiv:1808.09492*.

Simon Ostermann, Michael Roth, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. 2018. Zemeval-2018 task 11: Machine comprehension using commonsense knowledge. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 747–757. Association for Computational Linguistics.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

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

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

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.

Daniil Sorokin and Iryna Gurevych. 2018. Modeling semantics with gated graph neural networks for knowledge base question answering. *arXiv preprint arXiv:1808.04126*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*, pages 4444–4451.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. DREAM: A challenge dataset and models for dialogue-based reading comprehension. *CoRR*, abs/1902.00164.

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

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

Liang Wang, Meng Sun, Wei Zhao, Kewei Shen, and Jingming Liu. 2018a. Yuanfudao at semeval-2018 task 11: Three-way attention and relational knowledge for commonsense machine comprehension. *arXiv preprint arXiv:1803.00191*.

Minjie Wang, Lingfan Yu, Quan Gan, Da Zheng, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Junbo Zhao, Haibin Lin, Chao Ma, Damon Deng, Qipeng Guo, Hao Zhang, Jinyang Li, Alexander J Smola, and Zheng Zhang. 2018b. Deep graph library.

Bishan Yang and Tom Mitchell. 2019. Leveraging knowledge bases in lstms for improving machine reading. *arXiv preprint arXiv:1902.09091*.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2018. From recognition to cognition: Visual commonsense reasoning. *arXiv preprint arXiv:1811.10830*.

Wanjun Zhong, Duyu Tang, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2018. Improving question answering by commonsense-based pre-training. *arXiv preprint arXiv:1809.03568*.