GrapeQA: GRaph Augmentation and Pruning to Enhance Question-Answering

Dhaval Taunk*, **Lakshya Khanna***, **Pavan Kandru***,
Vasudeva Varma, Charu Sharma and Makarand Tapaswi
IIIT Hyderabad, India
{dhaval.taunk,lakshya.khanna,siri.venkata}@research.iiit.ac.in
{vv,charu.sharma,makarand.tapaswi}@iiit.ac.in

Abstract

Commonsense question-answering (QA) methods combine the power of pre-trained Language Models (LM) with the reasoning provided by Knowledge Graphs (KG). A typical approach collects nodes relevant to the QA pair from a KG to form a Working Graph (WG) followed by reasoning using Graph Neural Networks (GNNs). This faces two major challenges: (i) it is difficult to capture all the information from the QA in the WG, and (ii) the WG contains some irrelevant nodes from the KG. To address these, we propose GrapeQA with two simple improvements on the WG: (i) Prominent Entities for Graph Augmentation identifies relevant text chunks from the QA pair and augments the WG with corresponding latent representations from the LM, and (ii) Context-Aware Node Pruning removes nodes that are less relevant to the QA pair. We evaluate our results on OpenBookQA, CommonsenseQA and MedQA-USMLE and see that GrapeQA shows consistent improvements over its LM + KG predecessor (QA-GNN in particular) and large improvements on OpenBookQA.

1 Introduction

2 Introduction

Answering questions is a challenging NLP problem as it involves understanding the question context and sifting through relevant information to identify the answer. Question-answering models have evolved from rule-based (Kahaduwa et al., 2017) to RNN-based sequence models (Meng et al., 2017) and now to Transformer-based Language Models (LM) such as RoBERTa-large (Liu et al., 2019). However, commonsense question-answering adds a layer of complexity as the model needs to reason about questions relating diverse topics, making the task challenging for LMs that may not have seen something similar in the pre-training data.

While LMs capture the implicit patterns and contextual information within the data, KGs are able to capture explicit relations between the text entities. KGs such as Freebase (Bollacker et al., 2008), Wikidata (Vrandečić, 2012), or ConceptNet (Speer et al., 2017) store knowledge in the form of graph triplets (topic-relationship-topic) and are well suited for Graph Neural Networks (GNNs), *e.g.* (Welling and Kipf, 2016). Thus, commonsense QA in particular has attracted interest in combining LMs and KGs with the reasoning ability of GNNs (Lin et al., 2019; Yasunaga et al., 2021).

Most works on LM + KG extract a sub-graph or Working Graph (WG) from the KG based on concepts mentioned in the QA pair (Lin et al., 2019; Feng et al., 2020; Yasunaga et al., 2021) and focus on improving reasoning. For example, Lin et al. (2019) propose a graph network to score answers while Feng et al. (2020) focus on a multi-hop message passing framework that allows each node to attend to multi-hop neighbors in a single layer, combining interpretable path-based reasoning with scalable GNNs. Yasunaga et al. (2021) improve the extracted WG through a relevance scoring mechanism followed by joint reasoning and Zhang et al. (2022) fuse information from both the modalities (LM, KG) by mixing their tokens and nodes.

Our emphasis with GrapeQA lies in improving the working graph (WG) with two simple ideas. (i) We augment the WG with useful information from the question-answer pair reducing the burden on a single QA context node used in previous works.(discussed in 3.2) (ii) Instead of keeping all nodes of the WG, or simply scoring relevance, we drop less relevant information (nodes) from the WG simplifying the graph reasoning process. The improvements to the WG are combined with the reasoning process of QA-GNN (Yasunaga et al., 2021) and evaluated on three datasets, where we see especially large improvements on domain-specific OpenBookQA.(discussed in 3.2)

*These authors contributed equally to this work.

3 GrapeQA Methodology

We briefly describe the QA-GNN approach before our graph augmentation and pruning strategies.

3.1 LM + KG: QA-GNN as a case study

The objective of QA-GNN (Yasunaga et al., 2021) is to use both LM and KG for commonsense QA tasks. Each multiple-choice QA consists of a question q and O answer options $\{a_o\}_{o=1}^{O}$ where only one is correct. We create one Working Graph (WG) per answer option and reason over the graph to produce a score. During training, cross-entropy loss is applied to scores of all answer options while we pick the highest scoring answer for inference.

We discuss the WG creation process starting with the KG. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the KG with $\mathcal{V}$ nodes and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ with $\mathcal{R}$ relation types. For a given question-answer pair $[q; a_o]$, all nodes in the KG may not be relevant. Hence, *Question / Answer entity nodes*, referred as q_{KG} or a_{KG}, that have some text matching with the question q or answer option a_o are picked. Indirect relations between Question and Answer entity nodes are captured through common neighbors (2-hop away) by including them as *Extra nodes* s_{KG}. The sub-graph $\mathcal{G}_{\text{sub}}$ is formed together with the edges in $\mathcal{E}$ that connect the chosen KG nodes. In summary, the nodes of the sub-graph are $\{q_{\text{KG}}\} \cup \{a_{\text{KG}}\} \cup \{s_{\text{KG}}\}$.

Next, a relevance scoring mechanism is used to prune irrelevant nodes that may appear in the sub-graph. Scores are computed by encoding the *QA context* (concatenated question and answer option text) and node label using an LM followed by a linear projection. The relevance score influences the node representation in the sub-graph. Finally, to create the Working Graph $\mathcal{G}_w$, *QA context* is added as a node to the sub-graph and connected with other nodes using a new edge type.

Question, Answer, and Extra nodes in $\mathcal{G}_w$ are initialized by creating sentences based on triplets from the KG, feeding them to a pretrained LM, and average pooling over relevant tokens (see (Feng et al., 2020) for details). The QA context node is initialized as $\mathbf{z}$, an encoding of the $[q; a_o]$ text using an LM. To perform reasoning, a relation type aware Graph Network is adopted. The output representations for all nodes are pooled and added to the LM's original encoding of the QA context. Finally, an MLP is used to predict a score for the correctness of the answer option. Fig. 1 illustrates QA-GNN along with our proposed modifications. Additional details of QA-GNN are in App. B.

3.2 Graph Augmentation and Pruning

GrapeQA proposes two improvements to the WG and corresponding adaptations to QA-GNN. We overcome the limited capacity of the WG to exchange useful information between the QA context and the KG with Prominent Entities for Graph Augmentation (PEGA) that introduces additional nodes from the QA pair to the WG. We also propose QA-Context-Aware Node Pruning (CANP), a pruning method that removes least relevant nodes.

Prominent Entities for Graph Augmentation (PEGA). Graph augmentation begins by extracting noun phrase chunks c from the question and answer pair $[q; a_o]$. We use Spacy's (Honnibal et al., 2020) *noun* chunk extractor f_{ext} to obtain

$$\mathcal{V}' = \{c \mid c \in f_{\text{ext}}([q; a])\}\,. \tag{1}$$

The QA context is fed as input to the LM and representations of all the sub-word tokens are obtained. Each extracted noun phrase is represented by averaging over the embeddings of its sub-word tokens. As part of augmentation, these *noun chunks nodes* ($\mathcal{V}'$) are added as new nodes of type n to the working graph $\mathcal{G}_w$. *Noun chunk nodes* also have two types of edges: r_{no} between all the new ($\mathcal{V}'$) and old $\mathcal{G}_w$ nodes, and r_{nn} among the noun chunks themselves resulting in an augmented WG, $\mathcal{G}'_w$:

$$\mathcal{E}' = \{\mathcal{V}' \times r_{nn} \times \mathcal{V}'\} \cup \{\mathcal{V}' \times r_{no} \times \mathcal{V}\}\,, \tag{2}$$

$$\mathcal{G}'_w = (\mathcal{V} \cup \mathcal{V}', \mathcal{E} \cup \mathcal{E}')\,. \tag{3}$$

QA Context-Aware Node Pruning (CANP) aims to remove the less relevant nodes from the WG. Our intuition is that some Extra nodes (i.e. 2-hop neighbors from the KG which do not match the QA text) may be less relevant to the QA as compared to the Question / Answer entity nodes.

To perform pruning, we first associate and cluster Extra nodes with *Answer entity nodes*. CANP is only applied when there are more than one Answer entity nodes. Recall that the WG is created for one answer option (or one QA pair) and the number of Answer entity nodes (and clusters) depends on the number of nodes with text similar to the answer option in the KG. Similar to relevance scoring in QA-GNN, we calculate the relevance score for each Extra node s_{KG} against each Answer entity

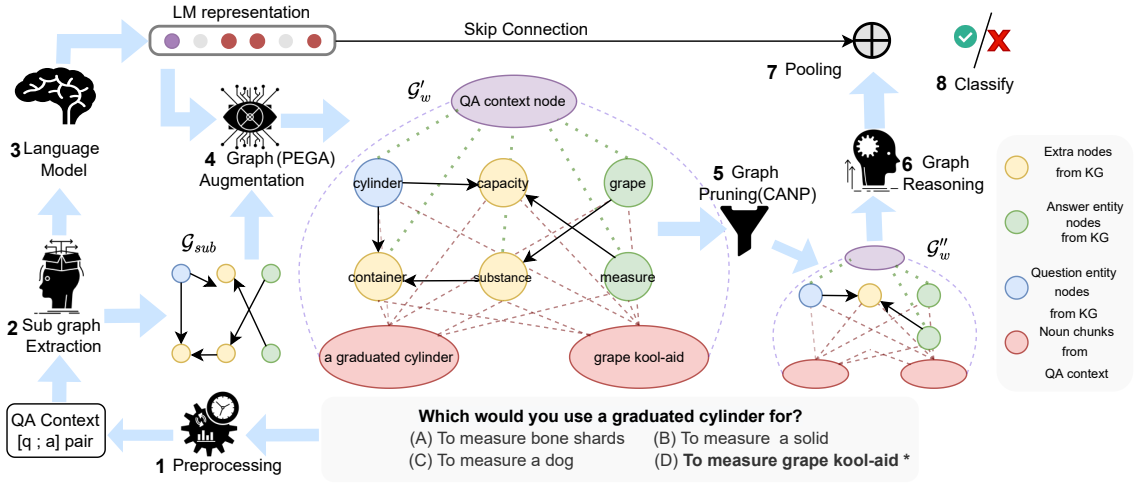

Figure 1: Method overview showing the approach to score the question with each answer option. GrapeQA improves QA-GNN (Yasunaga et al., 2021) by augmenting the Working Graph with additional nodes that capture information from the QA pair (step 4: PEGA) and then pruning the graph to remove the least relevant nodes (step 5: CANP).

a_{KG} by encoding the concatenated text of the QA pair, the Answer entity, and the Extra node.

$$\psi_{sa}^{\text{KG}} = f_{\text{head}}\left(\text{LM}\left([\text{text}(\mathbf{z}); \text{text}(\mathbf{a}_{\text{KG}}); \text{text}(\backslash)]\right)\right), \quad (4)$$

where $\text{text}(\cdot)$ corresponds to the node's label text: $[q; a_o]$ pair for $\mathbf{z}$, Extra node's label s_{KG} for \, and the Answer entity label a_{KG} for $\mathbf{a}_{\text{KG}}$. Thus, each Extra node s_{KG} is assigned to the cluster $\mathcal{V}_x$ corresponding to the highest relevance score,

$$\mathcal{V}_x = \{s_{\text{KG}} \,|\, x = \arg\max_{a_{\text{KG}}} \psi_{sa}^{\text{KG}}\}. \quad (5)$$

We compute the average relevance score for each cluster and identify the least relevant cluster $\mathcal{V}_r$ as

$$\psi_x^{\text{KG}} = \sum_{s_{\text{KG}} \in \mathcal{V}_x} \psi_{sx}^{\text{KG}} / |\mathcal{V}_x|, \quad (6)$$

$$\mathcal{V}_r = \mathcal{V}_x \text{ s.t. } r = \arg\min_x \psi_x^{\text{KG}}. \quad (7)$$

Finally, we remove the cluster with lowest average relevance score from the WG before continuing with graph-based reasoning. The PEGA augmented WG can be pruned as

$$\mathcal{G}''_w = (\mathcal{V} \cup \mathcal{V}' - \mathcal{V}_r, \mathcal{E} \cup \mathcal{E}' - \{\mathcal{V}_r \times R \times \mathcal{V}\}). \quad (8)$$

4 Experiments

We evaluate GrapeQA on three QA datasets:
1. CommonsenseQA (CSQA) is 5-way multiple-choice QA (MC-QA) dataset of 12,102 questions that requires commonsense reasoning to answer questions. We use standard splits (Lin et al., 2019) and report results on the in-house test (IHtest).
2. OpenBookQA (OBQA) is a 4-way MC-QA dataset of 5,957 questions based on elementary science knowledge; splits by Mihaylov et al. (2018).
3. MedQA-USMLE is a 4-way MC-QA dataset based on biomedical and clinical knowledge and has 12,723 questions from United States Medical License Exams, with splits by Jin et al. (2021).

Table 1 presents node counts in the WG for the above datasets while Table 7 (App. C) shows a small overlap between noun chunk and KG nodes.

Implementation & training details. The LM adopted in our work is RoBERTa-large (Liu et al., 2019) for CSQA and OBQA, and SapBERT (Liu et al., 2020) for MedQA. ConceptNet (Speer et al., 2017) is our KG for generating the WG in CSQA and OBQA. For MedQA, we use the graph constructed by QA-GNN (Yasunaga et al., 2021). Our model consists of an LM and a GNN with dim 200. RADAM optimizer is used with a learning rate of 10^{-5} for the LM and 10^{-3} for the GNN. OBQA & MedQA are trained for 50 epochs with a batch size of 128 and CSQA for 20 epochs with a batch size of 64. All models are a single run trained on 2 RTX 2080 Ti GPUs and take about 28 hours for

Table 1: Average number of nodes of each type in WGs.

Node Type	OBQA	CSQA	MedQA
Question entity q_{KG}	6.52	7.36	6.1
Answer entity a_{KG}	2.79	2.05	0.55
Extra nodes s_{KG}	107.17	112.04	20.82
Noun chunk nodes $\mathcal{V}'$	3.88	4.13	33.46

Table 2: Comparison of Accuracy between LM+KG methods on the OpenBookQA, CommonsenseQA (left) and MedQA (right).

Model	OBQA Test	CSQA IHTest
RGCN (Schlichtkrull et al., 2018)	62.45	68.4
GconAttn (Wang et al., 2019)	64.75	68.6
RN (Santoro et al., 2017)	65.20	69.1
MHGRN (Feng et al., 2020)	66.85	71.1
GreaseLM (AristoRoBERTa) (Zhang et al., 2022)	84.8	74.05
QA-GNN (RoBERTa-large) (Yasunaga et al., 2021)	67.80	73.4
GrapeQA: CANP (**Ours**)	66.20	**74.94**
GrapeQA: PEGA (**Ours**)	82.0	73.41
GrapeQA: PEGA+CANP (**Ours**)	**90.0**	74.05

MedQA	
Model	**Test**
BERT-base (Devlin et al., 2019)	34.3
BioBERT-base (Lee et al., 2019)	34.1
RoBERTa-large (Liu et al., 2019)	35.0
BioBERT-large (Lee et al., 2019)	36.7
SapBERT (Liu et al., 2020)	37.2
GreaseLM (Zhang et al., 2022)	38.5
QA-GNN (Yasunaga et al., 2021)	38.0
GrapeQA: (PEGA) (**Ours**)	**39.51**

Table 3: PEGA Ablations: Impact of different noun chunk extraction methods on OBQA.

Noun chunk extraction method	Accuracy
20% random words	72.32
NLTK (Loper and Bird, 2002)	78.40
spaCy (Honnibal et al., 2020)	**82.00**

OBQA and 16 hours for CSQA and MedQA.

4.1 Comparisons with Baselines

We use accuracy as a metric and compare our results primarily against other works that also adopt LM + KG methods (see Table 2). GrapeQA builds on top of QA-GNN (for direct comparison) and improving the WG results in highest performance on OBQA & MedQA and comparable performance on CSQA. For a fair comparison, we use the same LM for all methods unless noted.

LM only methods tend to perform worse than the baseline QA-GNN. RoBERTa-large (Liu et al., 2019) for CSQA provides 72.1% while RoBERTa-largeand AristoRoBERTa (Clark et al., 2019) for OBQA show 64.80% and 77.8%, respectively. For MedQA, the LM only model results are also shown in Table 2 (right); we see that LMs trained on medical data (*e.g.* SapBERT (Liu et al., 2020)) outperform generic LMs on this domain-specific task. GrapeQA outperforms all these approaches.

OBQA. CANP applied to the original QA-GNN WG is unable to improve performance (-1.6%), probably because the WG is not rich. However, PEGA provides a 14.2% accuracy improvement over QA-GNN (82% vs. 67.8%). Interestingly, CANP when used together with PEGA boosts the accuracy to 90% (+22.2%); surpassing GreaseLM that uses an improved LM (AristoRoBERTa) and better integration of LM + KG by 5.2%. For the *domain-specific* OBQA, PEGA adds relevant information while CANP effectively cleans up irrelevant nodes resulting in large improvements.

MedQA. PEGA achieves an improvement of 1.5% over QA-GNN, and 1% over GreaseLM, the previous SoTA. A reason for the small improvement (compared to OBQA) could be that the WG for MedQA has fewer nodes (see Table 1). Additionally, the small number of Answer entity nodes in the WG also means that CANP is not applicable.

CSQA. On *generic commonsense* questions, the WG can have large amounts of irrelevant information that CANP can simplify. We see an improvement of 1.5% over QA-GNN when using CANP only. However, unlike OBQA, PEGA shows comparable performance to QA-GNN as it may lead to stuffing the WG with common terms (noun chunks) that do not provide discriminatory information. Nevertheless, CANP alone also improves over GreaseLM by 0.9% (all in absolute points).

4.2 Ablation experiments

Noun chunk extraction. While PEGA is an effective graph augmentation strategy, it relies on the noun chunk extraction method. We evaluate automatic noun chunk extraction methods spaCy and NLTK (see App. D for details) against a simple baseline that randomly adds 20% of the QA pair's words to the WG. Table 3 shows that extracting meaningful chunks is important and may lead to large performance change (on OBQA). Interestingly, even random chunks of the QA pair provides a 4.5% boost over QA-GNN that only includes one node to encode the entire QA context.

Number of GNN layers is often an important

hyperparameter. We show results for both the PEGA+CANP (Table 4) and PEGA-only (Table 5) models in Appendix A. Generally, 5 layers seem to work well for all settings, while the CSQA PEGA-only model shows better results with 4 layers.

5 Conclusion

We presented GrapeQA, an effective approach to integrate information from QA (LM) and KG for commonsense QA. We proposed two simple improvements to the working graph: PEGA, a graph augmentation that improves information flow between the QA and the KG; and CANP that prunes less relevant information. Our approach led to new SoTA results on three datasets OBQA, CSQA, and MedQA, with a large 22% increase on OBQA.

6 Ethical Impact

In order to support commonsense thinking, this study suggests a general method for fusing language models and external knowledge graphs. We rely on publicly available datasets and benchmarks and knowledge graphs for each experiment. We could not anticipate any immediate social ramifications or ethical concerns as we neither amplify existing bias in the data nor do we inject any social or ethical bias into the model.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Peter Clark, Oren Etzioni, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, Niket Tandon, Sumithra Bhakthavatsalam, Dirk Groeneveld, Michal Guerquin, and Michael Schmitz. 2019. From 'F' to 'A' on the N.Y. Regents Science Exams: An Overview of the Aristo Project. *arXiv:1909.01958*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

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 *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1295–1309.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python. *10.5281/zenodo.1212303*.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Hasangi Kahaduwa, Dilshan Pathirana, Pathum Liyana Arachchi, Vishma Dias, Surangika Ranathunga, and Upali Kohomban. 2017. Question answering system for the travel domain. In *Moratuwa Engineering Research Conference (MERCon)*, pages 449–454.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv:2005.00700*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*. OpenReview.net.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839.

Fangyu Liu, Ehsan Shareghi, Zaiqiao Meng, Marco Basaldella, and Nigel Collier. 2020. Self-alignment pretraining for biomedical entity representations. *arXiv:2010.11784*.

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. *CoRR*, abs/1907.11692.

Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. *CoRR*, cs.CL/0205028.

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

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

Yuanliang Meng, Anna Rumshisky, and Alexey Romanov. 2017. Temporal information extraction for question answering using syntactic dependencies in an lstm-based architecture. *arXiv:1703.05851*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv:1809.02789*.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Neural Information Processing Systems (NeurIPS)*.

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.

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

Denny Vrandečić. 2012. Wikidata: A new platform for collaborative data collection. In *International Conference on World Wide Web (WWW)*, pages 1063–1064.

Peifeng Wang, Nanyun Peng, Filip Ilievski, Pedro Szekely, and Xiang Ren. 2020. Connecting the dots: A knowledgeable path generator for commonsense question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4129–4140, Online. Association for Computational Linguistics.

Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, et al. 2019. Improving natural language inference using external knowledge in the science questions domain. In *AAAI Conference on Artificial Intelligence*, pages 7208–7215.

Max Welling and Thomas N Kipf. 2016. Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*.

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 *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022. GreaseLM: Graph REASoning enhanced language models. In *International Conference on Learning Representations (ICLR)*.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2019. FreeLB: Enhanced Adversarial Training for Natural Language Understanding. *arXiv:1909.11764*.

Appendix

We first present additional results in Appendix A followed by a detailed explanation of QA-GNN in Appendix B. Appendix C provides some statistics for the datasets and working graphs, while Appendix D presents details of noun chunk extraction methods used in PEGA.

A Additional Results

Number of GNN layers ablations. Tables 4 and 5 show ablation studies by varying the number of GNN layers over PEGA+CANP and PEGA-only respectively. 5 layer GNNs seem to be a suitable for both methods, while CSQA with PEGA-only shows highest performance with 4 layers.

CANP is not necessary on MedQA. Table 1 of the main paper shows the average number of nodes of different types in a WG. The number of extra concept nodes is much higher than the QA concept nodes except in the MedQA dataset. This makes it necessary to prune these nodes to keep only the relevant ones. In case of MedQA since the number of extra nodes in WG are already quite low, and the nodes from the KG are often meaningful (domain-specific) we do not perform CANP pruning.

Results on CSQA. Table 6 shows the results of our model on the official test set for CommonsenseQA. We compare our results with other existing approaches, both using powerful LMs (e.g., UnifiedQA) or LM+KG methods (QA-GNN, GreaseLM, etc.). Unfortunately we were unable to evaluate our best performing model on the in-house test set (GrapeQA: CANP-only) due to limited number of submissions indicated for the evaluation. Even on the in-house test set, we see no performance change between PEGA-only and QA-GNN (73.41% vs. 73.4%) while a $\pm 1\%$ variation exists due to random seeds.

Table 4: Impact of the number of GNN layers using the PEGA+CANP model.

	Accuracy	
#layers	**OBQA**	**CSQA**
4	88.38	72.60
5	**90.00**	**74.05**
6	88.96	71.88

Table 5: Impact of the number of GNN layers using the PEGA only model.

	Accuracy	
#layers	**OBQA**	**CSQA**
4	83.20	**74.62**
5	**82.00**	73.41
6	81.40	73.17

Table 6: Comparison on CommonSenseQA official test set using RoBERTa-largemodel. The best result is in **bold** and second best is underlined. Due to limited entries for evaluation, we were unable to evaluate our best method on CSQA: CANP-only. *UnifiedQA has 11B parameters and is about 30x larger than QA-GNN and our model and is trained on much more data.

Model	**Test Acc.**
RoBERTa (Liu et al., 2019)	72.1
RoBERTa + FreeLB (ensemble) (Zhu et al., 2019)	73.1
RoBERTa + HyKAS (Ma et al., 2019)	73.2
RoBERTa + KE (ensemble)	73.3
RoBERTa+KEDGN (ensemble)	74.4
XLNet+GraphReason (Lv et al., 2019)	75.3
RoBERTa+MHGRN (Feng et al., 2020)	75.4
Albert+PG (Wang et al., 2020)	75.6
QA-GNN (Mihaylov et al., 2018)	76.1
Albert (ensemble) (Lan et al., 2020)	76.5
UnifiedQA* (Khashabi et al., 2020)	79.1
GrapeQA (PEGA) (**Ours**)	**73.5**

B QA-GNN Method Details

We provide further details of the question-answering procedure adopted by QA-GNN (Yasunaga et al., 2021).

B.1 Relevance Scoring

Extracting a sub-graph by selecting few hop neighbors adds many irrelevant nodes to the sub-graph. QA-GNN proposes a relevance scoring mechanism to add an "importance score" to the initial embedding of concept nodes. This helps the GNN to focus on nodes with high relevance score while performing graph reasoning.

$$\boldsymbol{\rho}_v = f_{\text{head}}\left(f_{\text{enc}}\left([\text{text}(\mathbf{z}); \text{text}(\mathbf{v})]\right)\right) , \quad (9)$$

$$\rho_t = f_\rho\left(\boldsymbol{\rho}_v\right) . \quad (10)$$

The text of each concept node $\text{text}(\mathbf{v})$ is concatenated with the QA-pair (referred to as $\text{text}(\mathbf{z})$). The LM encoding and following an MLP head produces an embedding $\boldsymbol{\rho}_v$ that is converted into a relevance score ρ_v using an MLP f_ρ. Nodes with a score lower than a threshold are discarded.

B.2 Node and Relation Types

QA-GNN constructs a working graph which is heterogeneous and multi-relational. It uses a node

type (u) aware and relation type (r) aware iterative message passing network to reason over it. Different node types are represented using embeddings $\mathbf{u}$. These include QA context, Question entity, Answer entity and Extra nodes. Whereas, edge embeddings $\mathbf{e}$ include relations in KG and two new relation types between the QA context node and KG entity nodes.

Node and relation types are embedded using MLPs f_u and f_r respectively,

$$\mathbf{u}_t = f_u\left(u_t\right) , \tag{11}$$

$$\mathbf{r}_{st} = f_r\left(\mathbf{e}_{st}, u_s, u_t\right) . \tag{12}$$

The message from the source to target node is constructed by concatenating the source node and type representations along with the relation embedding from the source to target and projecting it (to node embedding dimension) using the MLP f_m.

$$\mathbf{m}_{st} = f_m\left(\mathbf{h}_s^{(\ell)}, \mathbf{u}_s, \mathbf{r}_{st}\right) . \tag{13}$$

B.3 Message Passing

Node representations are updated at each layer using the following attention mechanism

$$\mathbf{q}_s = f_q\left(\mathbf{h}_s^{(\ell)}, \mathbf{u}_s, \boldsymbol{\rho}_s\right) , \tag{14}$$

and

$$\mathbf{k}_t = f_k\left(\mathbf{h}_t^{(\ell)}, \mathbf{u}_t, \boldsymbol{\rho}_t, \mathbf{r}_{st}\right) . \tag{15}$$

The query $\mathbf{q}_s$ and key $\mathbf{k}_t$ vectors of the source and target nodes are computed using the node representation $\mathbf{h}$, the node type embedding $\mathbf{u}$ and relevance score embeddings $\boldsymbol{\rho}$. Finally, we score attention α_{st} as

$$\alpha_{st} = \frac{\exp\left(\gamma_{st}\right)}{\sum_{t' \in \mathcal{N}_s \cup \{s\}} \exp\left(\gamma_{st'}\right)}, \ \gamma_{st} = \frac{\boldsymbol{q}_s^\top \boldsymbol{k}_t}{\sqrt{D}} . \tag{16}$$

The attention weights for messages from source to target α_{st} are calculated using $\mathbf{q}$ and $\mathbf{k}$ vectors. Finally, we aggregate messages and update the node representation as

$$\boldsymbol{h}_t^{(\ell+1)} = f_n\left(\sum_{s \in \mathcal{N}_t \cup \{t\}} \alpha_{st} \boldsymbol{m}_{st}\right) + \boldsymbol{h}_t^{(\ell)} . \tag{17}$$

Table 7: Number of unique nodes across all WG of the dataset. Even though more nodes are added from the KG on average (see Table 1), they are not all unique across the dataset and result in a smaller count.

Dataset	Noun chunk nodes	Nodes from KG	Overlapping nodes
OBQA	14470	7506	1958
CSQA	23881	12485	4023
MedQA	69370	2753	1268

Table 8: Average number of words in the question q and answer option a_o for the different datasets.

	Question	Answer
OBQA	13.5	2.8
CSQA	13.8	1.5
MedQA	116.2	3.6

C Working Graph Statistics

Given a question and corresponding answer option, KG nodes with matching text entities are identified. These matched nodes along with the Extra nodes that fall in 2-hop paths from them form the sub-graphs for each $[q; a]$ pair. Working Graphs are constructed by joining these sub-graphs with QA context nodes initialized with the representation from LM. In each Working Graph, the QA context node is connected to all the concept nodes in it which are extracted from the KG.

Node counts. Table 1 in the main paper shows the number of nodes added to the WG on average. We see that general KGs (ConceptNet) afford a large number of extra nodes (100+) while MedQA with a smaller KG only adds a few extra nodes (∼20). The large number of noun chunks added in the MedQA is explained by the fact that the questions in MedQA are very large as they include patient's description. Table 8 presents the average number of words in the question and answer option.

Noun chunks are unique. Table 7 shows the number of unique nodes present in each dataset. It can be observed that the total number of *unique* nodes selected from the KG is low as compared to the total number of unique noun chunk nodes extracted. Even though Table 1 shows that a large number of nodes are added to the graph, they are not all unique. Thus, even if the average number of noun chunk nodes for each WG are low, they are more diverse compared to nodes from KG. A small overlap between noun chunk nodes and nodes from the

KG indicates that this way of constructing the WG may provide better opportunity for graph reasoning to exchange information effectively between the QA (LM) and the KG.

D Noun chunk extraction methods

SpaCy is a Python and Cython programming language-based open-source software library for sophisticated natural language processing[1]. The library is distributed under the MIT licence.

In our experiments, we used `en_core_web_sm` package which provides functionalities like tok2vec, tagger, parser, attribute_ruler, lemmatizer, ner. We have used the noun chunk parser technique for extracting the noun chunks.

NLTK In order to work with human language data, Python programs can be built using the NLTK framework. NLTK offers simple access to more than 50 corpora and lexical resources, including WordNet, as well as a number of text processing libraries for categorization, tokenization, stemming, tagging, parsing, and semantic reasoning.

In our implementation, we first tokenize the input sentence using NLTK's `word_tokenizer`. Then to extract the noun chunks, the POS tagger of NLTK is used.

[1] https://spacy.io/