# Fair comparison of knowledge graphs for question answering

Anonymous ACL submission

## Abstract

Knowledge graphs are commonly used as sources of information in question answering. Models often combine pre-trained text encoders with a graph encoder to use this information to increase accuracy. However, the way that these two types of model interact is not clear. Here we show that, when provided with graph information for a random question, two recent models exhibit no significant change in performance. These models cannot therefore be used to obtain graph-structured explanations, or to compare the relevance of a particular knowledge graph to a dataset. We perform two model ablations and show that the resulting model is more responsive to variation in graph input, and so can be used for gathering explanations and measuring KG-dataset fit. We also show that uncontrollable nondeterminism can cause significant changes in results, and highlight the importance of statistical testing of these models.

## 1 Introduction

A question answering (QA) model must have access to relevant information that it can manipulate to provide an answer. A common approach is to store latent representations of this in the parameters of a Transformer model (Vaswani et al., 2017; Devlin et al., 2019). However, these models are for the most part uninterpretable.

Alternatively, this information can be stored symbolically, for instance in a knowledge graph. Approaches of this type typically select a subset of the graph, encode it, and combine it with an encoding of the question text to get a final representation, which is then used for selecting an answer (K M et al., 2018; Mihaylov and Frank, 2018; Lin et al., 2019; Feng et al., 2020; Yasunaga et al., 2021). An advantage of explicitly stating facts is that they provide a medium in which the model can output explanations for questions. These can be obtained, for example, by using graph-based interpretability techniques (e.g. Schlichtkrull et al., 2020) with models that use graph neural networks (GNNs) to encode the graph. The result is a set of triples which model developers can use to verify that only plausible facts are used in the inference process.

MHGRN (Feng et al., 2020) and QA-GNN (Yasunaga et al., 2021) are two contemporary models that combine a GNN-based graph encoder with a text encoder, which is usually instantiated as Roberta-Large (Liu et al., 2019). Before the models are run, a relevant knowledge graph must be identified and a subset extracted for use as input. Bauer and Bansal (2021) propose a method for choosing between different knowledge graphs for a task. They provide BERT (Devlin et al., 2019) with data from different graphs in addition to the question context, and select the graph that gives the highest performance.

In this paper we investigate the impact of using pre-trained models like BERT as a component in tasks involving knowledge graphs. We test MHGRN and QA-GNN under two input data conditions to evaluate how much the graph encoder and text encoder components contribute to question answering accuracy. We find that, under normal model conditions, swapping the graph input from useful to unhelpful data results in no significant change in performance. This result suggests that the text encoder has a disproportionate influence on model performance, which raises doubts about the use of these models to obtain graph-based explanations.

We further evaluate these models under two model ablation conditions that are designed to reduce the capability of the language model to learn the task. In this scenario unhelpful graph data causes performance to drop, suggesting that the performance of these models is more closely tied to the quality of the graph input. Disentangling graph-related performance from text encoder performance in this way can be seen to be more "fair" to the KGs, and it is crucial when using graphs to output explanations. For the goal of evaluating KG-dataset fit fairly, we therefore prefer our analysis method over the alternative provided by Bauer and Bansal (2021).

Our contributions are as follows[1]:

- Highlight flaws in existing methodology for testing models with graph encoders;
- Describe model changes to make them suitable for use with explanation generation methods and for testing KG-dataset fit;
- Demonstrate the importance of multiple model runs and statistical testing when using GNNs.

## 2 Background

### 2.1 Evaluating knowledge graphs

Bauer and Bansal (2021) propose a method to decide which knowledge graph is the most useful for a given multiple-choice question answering dataset. They begin by identifying potentially relevant triples from the knowledge graph based on lexical overlap with a candidate answer and optionally also the question. One or more of these is then concatenated with the question and answer, and the resulting string passed through BERT to obtain a score. The knowledge graph which has facts that lead to the highest question answering accuracy across all questions is selected as the most useful graph for the dataset.

Although this experimental setup is successful in finding the knowledge graph that is most useful for a given dataset, it does so in the context of the information held latent in the parameters of BERT. The interaction between this information and the facts in the input is unknown, and so drawing conclusions about a match between a knowledge graph and a dataset in general is difficult. This approach is also limited to measuring the impact of a small number of facts simultaneously, due to input token length limitations. It may be desirable to measure the impact of multiple facts that are more useful together than individually. This is likely to be more important for scientific questions (where multiple facts must be combined to reach an answer (Jansen et al., 2016)) as opposed to factoid questions, where typically one or two hops suffice.

[1] We release our code to facilitate future work.

### 2.2 Graph encoding QA models

We test the behaviour of two recent models that combine text encoders and graph encoders: MHGRN (Feng et al., 2020) and QA-GNN (Yasunaga et al., 2021). The high-level operation of both models is comparable. Text representing the question context – a question with an answer candidate – is used to extract a subgraph of up to 200 nodes (called a schema graph) from the overall knowledge graph. An encoding of the question context is also obtained from a text encoder, which in both cases is instantiated as Roberta-Large. Both models employ a message passing GNN as the graph encoder, in which the embeddings for nodes are initialised with pretrained embeddings.

MHGRN proposes a novel message passing GNN that builds a representation for each node by sampling paths from it. Each path is encoded into a message, which are combined using the textual embedding for structured relational attention, and then used to update the node embedding.

QA-GNN is a graph attention network (Veličković et al., 2018), where at each layer of the model the node embeddings are updated via message passing to immediate neighbours. A pseudo node is added to the graph to represent the textual context, and its embedding is initialised with the textual embedding. Through messages from this node, the data in the text embedding is spread across the graph, in contrast with MHGRN where the only use of the text embedding is to inform attention values.

Both models combine node embeddings via attentive pooling using the textual embedding, and calculate the representation for an answer candidate by concatenating this with the textual embedding.

## 3 Methodology

We design methods to test how MHGRN and QA-GNN use the schema graphs when scoring answers in multiple choice question answering.

The model conditions consist of the base model, plus two successive ablations: one where the textual embedding is completely removed from the final embedding, and a further one where the text encoder is not fine-tuned. Models are newly trained for each ablation scenario. Our reasoning is that, if we want to know how useful a graph is for a given task or want to output an explanation using the graph, the graph component of the network should be the only part which influences the scores given to each candidate. We reduce the ability of the language model to contribute information to the reasoning process almost completely by freezing its parameters: in our experiments Roberta-Large when not fine-tuned performs just above random.

We also test a scenario where the schema graphs are disabled by shuffling them across questions, in addition to the undisturbed scenario where each question has its proper schema graph. In the shuffled scenario we expect the schema graph to be irrelevant for the new question it is now applied to. As a result, if the models use the schema graph in a sensible way then performance should decrease. If not, the quality of explanations from these models is immediately discounted.

We use the same training hyperparameters as detailed by the respective authors, as detailed in appendix A. We change the maximum number of epochs to 70, and because we observed high variability within training across random seeds we use an early stopping patience of 30. We use ConceptNet (Speer et al., 2017) as the base knowledge graph, and evaluate on CommonsenseQA (CSQA) (Talmor et al., 2019) and Open Book Question Answering (OBQA) (Mihaylov et al., 2018), training separate models for each. We use standard dataset splits OBQA, and 'in house' splits for CSQA from prior work (Lin et al., 2019). We use Roberta-Large (Liu et al., 2019) as our text encoder, which we also use to initialise node embeddings in the GNN following Feng et al. (2020). We repeat each experiment with 10 different random seeds and report the mean accuracy.

## 4 Results

### 4.1 Text encoder influence

Table 1 gives our experimental results; supplementary details are provided in appendices B and C. Following Reimers and Gurevych (2017), we use the Kolmogorov-Smirnov test (Massey, 1951) to check whether the test score distributions for each pair of model-data setups are significantly different.

**Schema graph shuffling** We begin by investigating the impact of providing models with a schema graph that was extracted for a different question. We expect models that use the graphs in a meaningful way to perform poorly. However, there is no significant difference in performance for both MHGRN and QA-GNN when the model is provided with a schema graph for a random question versus the correct one. It is not clear therefore whether or how the graph component of the model is used; instead it is likely that it is the text encoder in both architectures that is responsible for the relatively high accuracies.

| | CSQA | | OBQA | |
|---|---|---|---|---|
| | *Reg.* | *Shuf.* | *Reg.* | *Shuf.* |
| QA-GNN | 70.26 | 69.72 | 62.98 | 65.24 |
| – *Embed.* | 64.68 | 60.68 | 52.36 | 53.66 |
| – *Train TE.* | 30.46 | 19.50 | 40.70 | 25.26 |
| MHGRN | 69.71 | 69.07 | 65.98 | 65.50 |
| – *Embed.* | 24.66 | 19.64 | 42.56 | 31.96 |
| – *Train TE.* | 24.45 | 19.76 | 41.04 | 36.00 |

Table 1: Average accuracy (10 random seeds) in two schema graph scenarios for each dataset: *regular*, where schema graphs are correctly paired with questions, and where the mapping is *shuffled*. '– *Embed.*' is the model ablation where the text encoder embedding is removed from the final score calculation. '– *Train TE.*' additionally freezes the text encoder weights.

We suggest that an appropriate model to measure the fit between a question answering dataset and a knowledge graph should, as far as possible, only score answer candidates based on the contents of the graph. We therefore turn our attention to the models without the text encoder embedding and with the text encoder weights frozen. The change in performance when using the different schema graphs is significant in three of the four (model, dataset) combinations ($p < 2e{-}5$, except the ablated MHGRN on OBQA). The fact that there is a significant difference between performance when using schema graphs of varying relevance suggests that this is an appropriate setup for investigating the impact of graph selection on question answering performance.

**Model ablation impact** For both models, datasets, and schema graphs, removing the text encoder embedding from final score calculation causes a significant reduction in score in all eight cases ($p < 0.02$). The reduction is larger for MHGRN, where performance drops to below random on CSQA, than for QA-GNN, where performance decreases by 7% on average. Considering only the *regular* schema graph, this suggests either that MHGRN is unable to learn how to use the graph with the text encoder embedding removed, or that the schema graph does not contain useful information for the task. However, because the model achieves 42.56% accuracy on OBQA in the same ablation scenario, we conclude that the second case is more likely.

The fact that QA-GNN performance drops less than MHGRN when the text encoder embedding is removed is likely explained by their different architectures. QA-GNN also includes the text encoder embedding itself in its graph encoder, whereas MHGRN only uses it to inform attentive pooling. It is therefore easier for the text encoder in the former model to learn the task. The significant ($p < 0.003$) change in accuracy for all four QA-GNN models when the text encoder weights are frozen appears to confirm that much of the performance is attributable to the text encoder, not the graph encoder and by extension the graph.

There is one anomalous change in MHGRN's performance, where accuracy increases from 31.96% to 36.00% when freezing the text encoder weights and using a shuffled schema graph on OBQA. As there is unlikely to be a meaningful way to combine nodes from a random schema graph for a question, the training signal for the text encoder as it performs attentive pooling is likely to be noisy. When the text encoder is frozen the influence of this noise is removed.

### 4.2 Uncontrollable nondeterminism

We perform additional experiments to investigate the impact of nondeterminism when updating sparse tensors using PyTorch Geometric (Fey and Lenssen, 2019).[2] This is due to float imprecision, and is impractical to control.[3] This only affects QA-GNN. We choose one random seed and run ten models[4] on CSQA in each model ablation scenario.

| | Mean | Min | Max | SD |
|---|---|---|---|---|
| QA-GNN | 69.95 | 68.90 | 70.75 | 0.58 |
| − *Embed.* | 69.07 | 67.20 | 70.19 | 1.05 |
| − *Train TE.* | 31.44 | 28.77 | 32.72 | 1.24 |

Table 2: Nondeterministic variation in QA-GNN performance on CSQA across 10 runs with the same random seed.

In all three model settings in table 2, there is a significant difference[5] ($p < 0.03$) between accuracy of the best and worst performing models. When comparing all pairs of models, in the − *Train LM* ablation there are two significantly different pairs ($p < 0.05$).[6] This variation highlights the importance of completing multiple runs when using non-deterministic operations.

Assuming that variability in the graph encoder is roughly constant, the lower the standard deviation of accuracy, the larger the influence of the text encoder on the predictions, and therefore accuracy. As such, the high standard deviation in the − *Train LM* ablation provides further suggests that this is an appropriate scenario for evaluating knowledge graph-to-dataset match, at the expense of requiring multiple runs.

## 5 Discussion

We have demonstrated that in MHGRN and QA-GNN, as the ability of the pre-trained text encoder to learn a task is curtailed, performance significantly decreases, suggesting that it was this that most contributed to high accuracy scores. This is reinforced by the fact that providing the unmodified models with graphs intended for other questions has no significant impact on performance. Models with this behaviour are unsuitable for use when providing explanations for questions.

We propose that a model where the text encoder weights are frozen, and the text embedding is not part of the final representation, is suitable both for providing explanations for questions and for evaluating KG-dataset fit. This is because performance is driven predominantly by the contents of the graph, rather than the text encoder.

Our results emphasise the importance of repeating experiments, especially when using GNNs that have uncontrollable nondeterminism, as this can cause significant differences in results. Particularly in light of this, it is crucial to perform statistical tests between the results of two types of model.

Further work is required on schema graph selection methods. Our results on CSQA suggest that the graphs obtained may be unsuitable, as the models which are driven most by graph contents do not perform well on this dataset. Extracting via just two hops in the graph is unlikely to yield sufficient data for science questions (Jansen et al., 2016).

[2]PyTorch Geometric uses PyTorch Scatter for this purpose. We use versions 1.6.0 and 2.0.8 respectively, with PyTorch 1.8.1, CUDA 11.4, and an Nvidia A100 GPU.

[3]See discussion `https://github.com/rusty1s/pytorch_scatter/issues/226`

[4]One of the original 10 models for '− *Embed.*' gave a significantly anomalous test accuracy of 31.35%. Such situations are easy to identify and remove, so we report statistics that use a further run.

[5]Monte Carlo permutation test with 1 million iterations.

[6]Bonferroni correction applied to account for 45 comparisons.

## References


Lisa Bauer and Mohit Bansal. 2021. Identify, Align, and Integrate: Matching Knowledge Graphs to Commonsense Reasoning Tasks. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2259–2272, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1295–1309, Online. Association for Computational Linguistics.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Peter Jansen, Niranjan Balasubramanian, Mihai Surdeanu, and Peter Clark. 2016. What's in an Explanation? Characterizing Knowledge and Inference Requirements for Elementary Science Exams. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2956–2965, Osaka, Japan. The COLING 2016 Organizing Committee. 00013.

Annervaz K M, Somnath Basu Roy Chowdhury, and Ambedkar Dukkipati. 2018. Learning beyond datasets: Knowledge graph augmented neural networks for natural language processing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 313–322, New Orleans, Louisiana. Association for Computational Linguistics.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2822–2832, Hong Kong, China. Association for Computational Linguistics.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2021. On the Variance of the Adaptive Learning Rate and Beyond. *arXiv:1908.03265 [cs, stat]*. 00707 arXiv: 1908.03265.

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:1907.11692*.

Frank J. Massey. 1951. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, 46(253):68–78.

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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391. Association for Computational Linguistics.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable Reader: Enhancing Cloze-Style Reading Comprehension with External Commonsense Knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832, Melbourne, Australia. Association for Computational Linguistics. 00008.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. 2020. Interpreting Graph Neural Networks for NLP With Differentiable Edge Masking. 00010.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Minneapolis, MN. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *arXiv:1710.10903 [cs, stat]*. 05507 arXiv: 1710.10903.

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 *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

## A Hyperparameters

**QA-GNN** All parameters optimised with RAdam (Liu et al., 2021). Batch size is 128. A maximum of 128 tokens are input to the text encoder, which is trained with learning rate $1e-5$ but frozen for the first 4 epochs. The 5-layer GNN has 200 dimensional embeddings and is trained with learning rate $1e-3$. Parameters have L2 weight decay of 0.01 applied.

**MHGRN** All parameters optimised with RAdam. Batch size is 32. A maximum of 128 tokens are input to the text encoder, which is trained with learning rate $1e-5$ but frozen for the first 3 epochs. The 1-layer GNN has 100 dimensional embeddings and is trained with learning rate $1e-3$. Each layer performs 3-hop message passing. Parameters have L2 weight decay of 0.01 applied.

## B Additional results

Standard deviations on the test set for each experiment are given in table 3, and development set scores in table 4. The average run time of these experiments on a Nvidia A100 GPU are shown in table 5, which correspond to the number of optimisation steps in table 6.

| | CSQA | | OBQA | |
|---|---|---|---|---|
| | *Reg.* | *Shuf.* | *Reg.* | *Shuf.* |
| QA-GNN | 1.02 | 0.53 | 3.03 | 2.90 |
| – *Embed.* | 1.42 | 2.26 | 10.12 | 6.75 |
| – *Train TE.* | 1.19 | 1.32 | 2.36 | 2.28 |
| MHGRN | 0.73 | 0.91 | 2.48 | 1.68 |
| – *Embed.* | 0.79 | 1.12 | 6.60 | 13.18 |
| – *Train TE.* | 0.68 | 1.31 | 2.37 | 10.11 |

Table 3: Standard deviation of test set score across 10 runs, corresponding to table 1.

## C Model-model comparison

We compare performance between QA-GNN and MHGRN when using the regular schema graphs. Both models differ mainly in their graph encoder architecture, and the comparison made by the later work (Yasunaga et al., 2021) involves using the unmodified models. Here, we compare the models along another dimension by isolating the graph encoders.

| | CSQA | | OBQA | |
|---|---|---|---|---|
| | *Reg.* | *Shuf.* | *Reg.* | *Shuf.* |
| QA-GNN | 76.11 | 75.82 | 65.36 | 68.08 |
| – *Embed.* | 74.13 | 70.76 | 56.78 | 59.02 |
| – *Train TE.* | 33.49 | 22.27 | 45.62 | 28.30 |
| MHGRN | 75.24 | 75.28 | 69.52 | 69.24 |
| – *Embed.* | 29.37 | 22.44 | 47.76 | 36.68 |
| – *Train TE.* | 29.07 | 22.29 | 44.82 | 42.08 |

Table 4: Average development set accuracy across 10 runs, used to select the test scores reported in table 1.

| | CSQA | | OBQA | |
|---|---|---|---|---|
| | *Reg.* | *Shuf.* | *Reg.* | *Shuf.* |
| QA-GNN | 3.29 | 2.92 | 2.12 | 2.07 |
| – *Embed.* | 4.21 | 4.06 | 1.96 | 2.07 |
| – *Train TE.* | 1.41 | 1.56 | 0.97 | 1.01 |
| MHGRN | 2.23 | 2.23 | 1.85 | 1.72 |
| – *Embed.* | 2.02 | 2.90 | 1.25 | 1.50 |
| – *Train TE.* | 2.04 | 2.72 | 1.27 | 1.45 |

Table 5: Average run time (in hours) for experiments in table 1.

When the text encoder is frozen and its embedding removed, there is no significant difference in performance on OBQA. This suggests that both models are able to use the information in the schema graph equivalently. For CSQA, the difference between the score is significantly different ($p < 2e-5$), although both models perform just above random. Further research is required to investigate the consistently lower performance in CSQA compared with OBQA when freezing the text encoder.

| | CSQA | | OBQA | |
|---|---|---|---|---|
| | *Reg.* | *Shuf.* | *Reg.* | *Shuf.* |
| QA-GNN | 3524 | 3162 | 2769 | 2683 |
| − *Embed.* | 4509 | 4348 | 2582 | 2718 |
| − *Train TE.* | 2539 | 2827 | 2204 | 2293 |
| MHGRN | 10,906 | 10,826 | 10,881 | 10,168 |
| − *Embed.* | 9922 | 13,992 | 7487 | 8975 |
| − *Train TE.* | 10,028 | 13,273 | 7564 | 8634 |

Table 6: Average number of optimisation steps for experiments in table 1.