Learning to Deceive Knowledge Graph Augmented Models via Targeted Perturbation

Mrigank Raman[1,*], **Siddhant Agarwal**[2,*], **Peifeng Wang**[3], **Xiang Ren**[3]
[1]Indian Institute of Technology, Delhi, [2] Indian Institute of Technology, Kharagpur,
[3]University of Southern California
`mt1170736@iitd.ac.in, agarwalsiddhant10@iitkgp.ac.in,`
`wang-hs17@mails.tsinghua.edu.cn,`
`{peifengw,chanaaro,xiangren}@usc.edu`

Abstract

Knowledge graphs (KGs) have helped neural-symbolic models improve performance on various knowledge-intensive tasks, like question answering and item recommendation. By using attention over the KG, such models can also "explain" which KG information was most relevant for making a given prediction. In this paper, we question whether these models are really behaving as we expect. We demonstrate that, through a reinforcement learning policy (or even simple heuristics), one can produce deceptively perturbed KGs which maintain the downstream performance of the original KG while significantly deviating from the original semantics and structure. Our findings raise doubts about KG-augmented models' ability to leverage KG information and provide plausible explanations.

1 Introduction

Recently, neural reasoning over symbolic knowledge graphs (KGs) has emerged as a popular paradigm in machine learning. Such neural-symbolic KG (NSKG) models have improved performance on a number of knowledge-intensive downstream tasks: for question answering (QA), the KG provides context about how a given answer choice is related to the question [1, 2, 3, 4]; for item recommendation [5, 6, 7, 8], the KG mitigates data sparsity and cold start issues. Furthermore, by using attention over the KG, such models can "explain" which KG information was most relevant for making a given prediction [1, 2, 5, 7, 9, 10]. Nonetheless, the process in which NSKG models reason about KG information is still not well understood. It is generally assumed that, like humans, NSKG models base their predictions on semantically meaningful chains of entities and relations in the KG and that this process is responsible for KG-associated performance gains [1, 2, 10, 11].

In this paper, we challenge the above assumption and question whether existing NSKG models actually use KGs in a faithful manner. We study this question primarily by measuring model performance when all edges in the KG have been perturbed. To perturb the KG, we propose a reinforcement learning (RL) based perturbation algorithm and four perturbation heuristics. Surprisingly, for NSKG models on both commonsense QA and item recommendation, we find that the KG can be extensively perturbed with little to no effect on performance. This raises doubts about the plausibility of KG-based explanations and the role of KGs in current NSKG models. To the best of our knowledge, we are the first to analyze the effects of targeted KG perturbations on NSKG model performance.

2 Problem Setting

Our goal is to investigate whether NSKG models faithfully use KGs for making predictions. Across various perturbation methods, we measure model performance when every edge in the KG has been perturbed. To quantify how much the KG has been perturbed, we also measure the similarity between

*Work done while MR, SA and HW interned remotely at USC. Code and data are available at `https://github.com/INK-USC/deceive-KG-models`.

original KG and perturbed KG. Assuming the original KG contains accurate information, if the perturbed KG's facts deviate significantly from the original KG's facts, then a faithful NSKG model should achieve worse performance with the perturbed KG than with the original KG.

Notation Let $\mathcal{F}_\theta$ be an NSKG model, and let $(X_{\text{train}}, X_{\text{dev}}, X_{\text{test}})$ be a dataset for some downstream task. We denote a KG as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ is the set of entities (nodes), $\mathcal{R}$ is the set of relation types, and $\mathcal{T} = \{(e_1, r, e_2) \mid e_1, e_2 \in \mathcal{E},\ r \in \mathcal{R}\}$ is the set of facts (edges) composed from existing entities and relations. Let $\mathcal{G}^{'} = (\mathcal{E}, \mathcal{R}', \mathcal{T}^{'})$ be the KG obtained after perturbing $\mathcal{G}$, where $\mathcal{R}^{'} \subseteq \mathcal{R}$ and $\mathcal{T}^{'} \neq \mathcal{T}$. Let $f(\mathcal{G}, \mathcal{G}^{'})$ be a function that measures similarity between $\mathcal{G}$ and $\mathcal{G}^{'}$. Let $g(\mathcal{G})$ be the downstream performance when evaluating $\mathcal{F}_\theta$ on X_{test} and $\mathcal{G}$. Also, let $\oplus$ denote the concatenation operation, and let $\mathcal{N}_L(e)$ denote the set of L-hop neighbors for entity $e \in \mathcal{E}$.

High-Level Procedure First, we train $\mathcal{F}_\theta$ on X_{train} and $\mathcal{G}$, then evaluate $\mathcal{F}_\theta$ on X_{test} and $\mathcal{G}$ to get the original performance $g(\mathcal{G})$. Second, we freeze $\mathcal{F}_\theta$, then perturb $\mathcal{G}$ to obtain $\mathcal{G}^{'}$. Third, we evaluate $\mathcal{F}_\theta$ on X_{test} and $\mathcal{G}^{'}$ to get the perturbed performance $g(\mathcal{G}^{'})$. Finally, we measure $g(\mathcal{G}) - g(\mathcal{G}')$ and $f(\mathcal{G}, \mathcal{G}^{'})$ to assess how faithful the model is. This procedure is illustrated in Figure 2 of Appendix A.1. We consider two downstream tasks: commonsense QA and item recommendation.

Commonsense QA Given a question x and a set of k possible answers $\mathcal{A} = \{y_1, ..., y_k\}$, the task is to predict a compatibility score for each (x, y) pair, such that the highest score is predicted for the correct answer. In commonsense QA, the questions are designed to require commonsense knowledge which is typically unstated in natural language, but more likely to be found in KGs [4]. Let $\mathcal{F}_\phi^{\text{text}}$ be a text encoder, $\mathcal{F}_\psi^{\text{graph}}$ be a graph encoder, and $\mathcal{F}_\xi^{\text{cls}}$ be a classifier, where $\phi, \psi, \xi \subset \theta$. Let $\mathcal{G}_{(x,y)}$ denote a subgraph of $\mathcal{G}$ consisting of entities mentioned in text sequence $x \oplus y$, plus their corresponding edges. We start by computing a text embedding $\mathbf{h}_{\text{text}} = \mathcal{F}_\phi^{\text{text}}(x \oplus y)$ and a graph embedding $\mathbf{h}_{\text{graph}} = \mathcal{F}_\phi^{\text{graph}}(\mathcal{G}_{(x,y)})$. After that, we compute the score for (x, y) as $S_{(x,y)} = \mathcal{F}_\xi^{\text{cls}}(\mathbf{h}_{\text{text}} \oplus \mathbf{h}_{\text{graph}})$. Finally, we select the highest scoring answer: $y_{\text{pred}} = \arg\max_{y \in A}\ S_{(x,y)}$. KG-augmented commonsense QA models vary primarily in their design of $\mathcal{F}_\psi^{\text{graph}}$. In particular, path-based models compute the graph embedding by using attention to selectively aggregate paths in the subgraph. The attention scores can help explain which paths the model focused on most for a given prediction [1, 2, 12].

Item Recommendation We consider a set of users $\mathcal{U} = \{u_1, u_2, ..., u_m\}$, a set of items $\mathcal{V} = \{v_1, v_2, ..., v_n\}$, and a user-item interaction matrix $\mathbf{Y} \in \mathbb{R}^{m \times n}$. If user u has been observed to engage with item v, then $y_{uv} = 1$ in the user-item interaction matrix; otherwise, $y_{uv} = 0$. Additionally, we consider a KG $\mathcal{G}$, in which $\mathcal{R}$ is the set of relation types in $\mathcal{G}$. In $\mathcal{G}$, nodes are items $v \in \mathcal{V}$, and edges are facts of the form (v, r, v'), where $r \in \mathcal{R}$ is a relation. For the zero entries in $\mathbf{Y}$ (i.e., $y_{uv} = 0$), our task is to predict a compatibility score for user-item pair (u, v), indicating how likely user u is to want to engage with item v. We represent each user u, item v, and relation r as embeddings $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{r}$, respectively. Given a user-item pair (u, v), its compatibility score is computed as $\langle \mathbf{u}, \mathbf{v} \rangle$, the inner product between $\mathbf{u}$ and $\mathbf{v}$. KG-augmented recommender systems differ mainly in how they use $\mathcal{G}$ to compute $\mathbf{u}$ and $\mathbf{v}$. Generally, these models do so by using attention to selectively aggregate items/relations in $\mathcal{G}$. The attention scores can help explain which items/relations the model found most relevant for a given prediction [5, 7].

3 RL-Based KG Perturbation

We propose an RL-based approach for perturbing the KG. Given a KG, $\mathcal{G}$, we train a policy to output a perturbed KG, $\mathcal{G}'$, such that original-perturbed KG similarity, $f(\mathcal{G}, \mathcal{G}^{'})$, is minimized, while downstream performance, $g(\mathcal{G}^{'})$, is maximized. Specifically, the RL agent is trained to perturb $\mathcal{G}$ via *relation replacement*, so we call our algorithm *RL-RR*. Because the agent is limited to applying $N = |\mathcal{T}|$ perturbations to $\mathcal{G}$, our RL problem is framed as a finite horizon Markov decision process. In the rest of this section, we introduce a metric for measuring semantic KG similarity; define the actions, states, and reward in our RL problem; and explain how RL-RR is implemented.

Aggregated Triple Score (ATS) To measure semantic similarity between two KGs, we propose ATS. Let $s_\mathcal{G}$ be an edge (triple) scoring function, such that $s_\mathcal{G}(e_1, r, e_2)$ measures how likely edge (e_1, r, e_2) is to exist in $\mathcal{G}$. Also, assume $s_\mathcal{G}$ has been pre-trained on $\mathcal{G}$ for link prediction. Then, ATS is defined as $f_{\text{ATS}}(\mathcal{G}, \mathcal{G}^{'}) = \frac{1}{|\mathcal{T}'|} \sum_{(e_1, r, e_2) \in \mathcal{T}'} s_\mathcal{G}(e_1, r, e_2)$, which denotes the mean $s_\mathcal{G}$ score across all edges in $\mathcal{G}'$. Intuitively, if a high percentage of edges in $\mathcal{G}'$ are also likely to exist in $\mathcal{G}$ (i.e., high ATS), then we can say that $\mathcal{G}'$ and $\mathcal{G}$ have high semantic similarity. Appendix A.3 has more details about $s_\mathcal{G}$.

Actions The action space consists of all possible relation replacements in $\mathcal{G}$, i.e., replacing $(e_1, r, e_2) \in \mathcal{T}$ with (e_1, r', e_2). Since having such a large action space poses computational issues, we decouple each action into a sequence of three *subactions* and operate instead in this smaller subaction space. Hence, a perturbation action at time step t would be $a_t = (a_t^{(0)}, a_t^{(1)}, a_t^{(2)})$. Namely, $a_t^{(0)}$ is sampling entity $e_1 \in \mathcal{E}$; $a_t^{(1)}$ is selecting edge $(e_1, r, e_2) \in \mathcal{T}$; and $a_t^{(2)}$ is selecting relation $r' \in \mathcal{R}$ to replace r in (e_1, r, e_2). To make the policy choose low-ATS perturbations, we further restrict the $a_t^{(2)}$ subaction space to be the K subactions resulting in the lowest ATS. Note that each $a_t^{(i)}$ is represented by its corresponding pre-trained TransE [13] entity, edge, or relation embedding in $\mathcal{G}$. Since these TransE embeddings are not updated by the perturbation policy, we use $a_t^{(i)}$ to refer to both the subaction and subaction embedding. Meanwhile, a_t does not have any representation besides its constituent subaction embeddings.

States The state space is the set of all $\mathcal{G}'$ with the same entities and connectivity structure as $\mathcal{G}$. Here, we make a distinction between state and state embedding. The state at t is the actual KG after t perturbation steps and is denoted as $\mathcal{G}_t$. The state embedding at t is a vector representation of $\mathcal{G}_t$ and is denoted as s_t. To match a_t, we also decouple s_t into *substate* embeddings: $s_t = (s_t^{(0)}, s_t^{(1)}, s_t^{(2)})$.

Reward The reward function pushes the policy to maximize downstream performance. For commonsense QA, higher reward corresponds to lower KL divergence between the predicted and true answer distributions. For item recommendation, we use dev AUC as the reward function.

3.1 DQN Architecture and Training

As described above, RL-RR is modeled as an action-subaction hierarchy. At the action (top) level, for t, the policy selects an action a_t given state s_t, then performs a_t on $\mathcal{G}_t$ to obtain $\mathcal{G}_{t+1}$. At the subaction (bottom) level, for index $i \in [0, 1, 2]$ within time step t, the policy selects a subaction $a_t^{(i+1)}$ given $s_t^{(i)}$ and, if any, previous subactions.

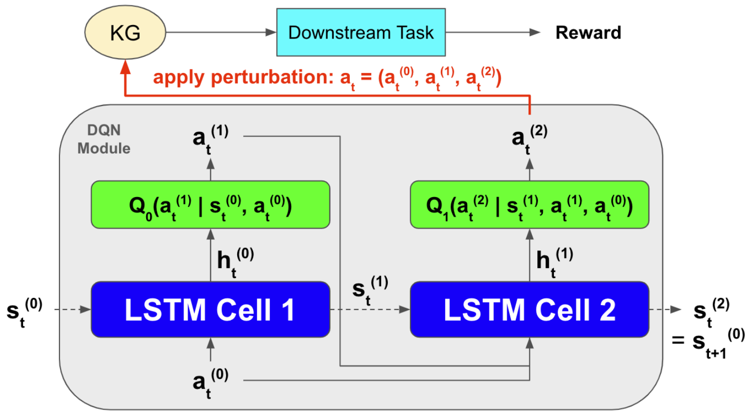


Figure 1: **DQN Architecture for RL-RR.**

At t, the policy takes as input the substate embedding $s_t^{(0)}$. One approach for computing $s_t^{(0)}$ would be to directly encode $\mathcal{G}_t$ with a graph encoder $\mathcal{F}^{\text{graph}}$, such that $s_t^{(0)} = \mathcal{F}^{\text{graph}}(\mathcal{G}_t)$ [14, 15, 16]. However, since we aim to assess graph encoders' ability to capture KG information, it would not make sense to use a graph encoder for KG perturbation. Instead, we use an LSTM [17] to update substate embeddings both within and across time steps, while jointly encoding substate and subaction embeddings. Observe that this means $s_t^{(i)}$ only implicitly captures KG state information via $a_t^{(i-1)}$, since the choice of each subaction is constrained precisely by which entities, edges, or relations are available in $\mathcal{G}_t$.

To train RL-RR, we use the DQN algorithm [18]. Abstractly, the goal of DQN is to learn a Q-function $Q(s_t, a_t)$, which outputs the expected reward for taking action a_t in state s_t. In practice, $Q(s_t, a_t)$ is decomposed into a sequential pair of sub-Q-functions: $Q_1(a_t^{(1)} | s_t^{(0)}, a_t^{(0)}) = \langle \text{MLP}(a_t^{(1)}), \text{MLP}(h_t^{(0)}) \rangle$ and $Q_2(a_t^{(2)} | s_t^{(1)}, a_t^{(0)}, a_t^{(1)}) = \langle \text{MLP}(a_t^{(2)}), \text{MLP}(h_t^{(1)}) \rangle$. MLP denotes the vector representation computed by a multi-layer perceptron, while $h_t^{(0)}$ and $h_t^{(1)}$ denote the respective LSTM encodings of $(s_t^{(0)}, a_t^{(0)})$ and $(s_t^{(1)}, [a_t^{(0)} \oplus a_t^{(1)}])$.

Figure 1 depicts the perturbation procedure at t. First, we either initialize $s_t^{(0)}$ with a trained embedding weight vector if $t = 0$, or set it to $s_{t-1}^{(2)}$ otherwise. Second, we uniformly sample $a_t^{(0)}$, which is encoded as $h_t^{(0)} = \text{LSTMCell}_1(s_t^{(0)}, a_t^{(0)})$. LSTMCell$_1$ also updates $s_t^{(0)}$ to $s_t^{(1)}$. Third, we compute $Q_1(a_t^{(1)} | s_t^{(0)}, a_t^{(0)})$, which takes $h_t^{(0)}$ as input and outputs $a_t^{(1)}$. Fourth, we encode $a_t^{(1)}$ as $h_t^{(1)} = \text{LSTMCell}_2(s_t^{(1)}, [a_t^{(0)} \oplus a_t^{(1)}])$. LSTMCell$_2$ also updates $s_t^{(1)}$ to $s_t^{(2)}$. Fifth, we compute

Table 1: **Comparison of Perturbation Methods on CSQA and MovieLens-20M.** For CSQA, we follow the standard protocol of reporting test accuracy on the in-house data split from [1], as the official test labels are not available. For MovieLens-20M, we report test AUC.

	CSQA				**MovieLens-20M**			
	RN		**MHGRN**		**KGCN**		**RippleNet**	
Methods	Acc (↑)	ATS (↓)	Acc (↑)	ATS (↓)	AUC (↑)	ATS (↓)	AUC (↑)	ATS (↓)
No KG	53.41	-	53.41	-	91.30	-	91.30	-
Original KG	56.87	0.940	57.21	0.940	96.62	0.960	96.62	0.960
Relation Swapping (RS)	53.42	0.831	53.42	0.831	**96.62**	0.678	**97.46**	0.678
Relation Replacement (RR)	53.42	0.329	52.22	0.329	96.50	0.413	97.45	0.413
Edge Rewiring (ER)	53.42	0.505	52.22	0.505	96.24	0.679	93.42	0.739
Edge Deletion (ED)	52.21	0.933	51.00	0.933	90.36	0.982	90.22	0.982
RL-RR	**55.21**	**0.322**	**55.52**	**0.314**	96.53	**0.401**	97.25	**0.268**

$Q_2(a_t^{(2)}|s_t^{(1)}, a_t^{(0)}, a_t^{(1)})$, which takes $h_t^{(1)}$ as input and outputs $a_t^{(2)}$. Note that $a_t^{(1)}$ and $a_t^{(2)}$ are selected ϵ-greedily during training and greedily during evaluation. Finally, using $a_t = (a_t^{(0)}, a_t^{(1)}, a_t^{(2)})$, we perturb $\mathcal{G}_t$ to get $\mathcal{G}_{t+1}$.

Ideally, for each t, we would evaluate $\mathcal{G}_{t+1}$ on the downstream task to obtain the reward. However, downstream evaluation is expensive, so we only compute reward every T time steps. Moreover, for the policy to generalize well, state embeddings $(s_{t-T+1}, ..., s_{t-1}, s_t)$ should not correlate with the order of actions $(a_{t-T+1}, ..., a_{t-1}, a_t)$. Thus, for every T time steps during training, we shuffle the last T actions after computing reward, then update the LSTM and sub-Q-functions with respect to the shuffled actions. Doing so encourages state embeddings to be invariant to action order.

4 Experiments

We test NSKG models on their ability to maintain downstream performance when the KG has been extensively perturbed. We report performance and ATS results for when all $|\mathcal{T}|$ edges in the KG are perturbed, averaged over three runs.

Baselines We consider several baselines for RL-RR, varying in how the KG is perturbed. In *No KG*, the model does not use any KG. In *Original KG*, the model uses the original KG. Additionally, we propose four KG perturbation heuristics: *Relation Swapping (RS)* randomly chooses two edges from $\mathcal{T}$ and swaps their relations. *Relation Replacement (RR)* randomly chooses an edge $(e_1, r_1, e_2) \in \mathcal{T}$, then replaces r_1 with another relation $r_2 = \text{argmin}_{r \in \mathcal{R}} s_{\mathcal{G}}(e_1, r, e_2)$. *Edge Rewiring (ER)* randomly chooses an edge $(e_1, r, e_2) \in \mathcal{T}$, then replaces e_2 with another entity $e_3 \in \mathcal{E} \setminus \mathcal{N}_1(e_1)$. *Edge Deletion (ED)* randomly chooses an edge $(e_1, r, e_2) \in \mathcal{T}$ and deletes it. For ED, perturbing all edges means deleting all but 10 edges. Figure 3 in Appendix A.2 depicts all five proposed perturbation methods.

Commonsense QA We consider the RN (with attentional path aggregation) [1, 12] and MHGRN [2] models, using BERT-Base [19] as the text encoder. We evaluate RN and MHGRN on the CSQA dataset [4], with ConceptNet [20] as the KG. In Table 1, for both RN and MHGRN, we see that RL-RR achieves slightly worse accuracy on CSQA than Original KG, while the heuristics generally perform on par with No KG. We also observe that the perturbed KGs produced by RS and ED have high ATS (i.e., semantic similarity to the original KG), while RR, ER, and RL-RR achieve relatively low ATS. In particular, among the five perturbation methods, RL-RR has the highest accuracy while also having the lowest ATS. This suggests that the model is not using the KG faithfully, since RL-RR achieves high performance despite extensively corrupting the original KG's semantic information.

Item Recommendation We experiment with the KGCN [7] and RippleNet [5] models on the MovieLens-20M dataset [21], using the item KG from [6]. In Table 1, for both KGCN and RippleNet, we find that relation-based perturbations tend to perform better, comparable to Original KG on MovieLens-20M. Again, ED achieves very high ATS, while other perturbation methods achieve more modest ATS. Notably, RL-RR is among the best in AUC, while also having the lowest ATS. Plus, even the RR achieves a relatively high AUC:ATS ratio. This provides further evidence that the model is not using the KG's semantic information in the manner we expect.

Additional Experiments We also conducted the following experiments: evaluating with other KG similarity metrics, evaluating on other datasets, measuring performance as a function of perturbation level, comparing to noisy baselines, asking humans to judge the readability and usability of NSKG model explanations, and validating our proposed KG similarity metrics. The descriptions and results of these additional experiments can be found in Appendix A.5.

References

[1] Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 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, 2019.

[2] Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. Scalable multi-hop relational reasoning for knowledge-aware question answering. *arXiv preprint arXiv:2005.00646*, 2020.

[3] Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering. In *AAAI*, 2020.

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

[5] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 417–426, 2018.

[6] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 968–977, 2019.

[7] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. Knowledge graph convolutional networks for recommender systems. In *The world wide web conference*, pages 3307–3313, 2019.

[8] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. Shine: Signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 592–600, 2018.

[9] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*, pages 151–161, 2019.

[10] Jingyue Gao, Xiting Wang, Yasha Wang, and Xing Xie. Explainable recommendation through attentive multi-view learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3622–3629, 2019.

[11] Weiping Song, Zhijian Duan, Ziqing Yang, Hao Zhu, Ming Zhang, and Jian Tang. Explainable knowledge graph-based recommendation via deep reinforcement learning. *ArXiv*, abs/1906.09506, 2019.

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

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

[14] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. *arXiv preprint arXiv:1806.02371*, 2018.

[15] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Proceedings of The Web Conference 2020*, pages 673–683, 2020.

[16] Yao Ma, Suhang Wang, Tyler Derr, Lingfei Wu, and Jiliang Tang. Attacking graph convolutional networks via rewiring. *arXiv preprint arXiv:1906.03750*, 2019.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

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

[20] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. *arXiv preprint arXiv:1612.03975*, 2016.

[21] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, 2016.

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

[23] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases, 2015.

[24] Jari Saramäki, Mikko Kivelä, Jukka-Pekka Onnela, Kimmo Kaski, and Janos Kertesz. Generalizations of the clustering coefficient to weighted complex networks. *Physical Review E*, 75(2):027105, 2007.

[25] Jukka-Pekka Onnela, Jari Saramäki, János Kertész, and Kimmo Kaski. Intensity and coherence of motifs in weighted complex networks. *Physical Review E*, 71(6):065103, 2005.

[26] Giorgio Fagiolo. Clustering in complex directed networks. *Physical Review E*, 76(2):026107, 2007.

[27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

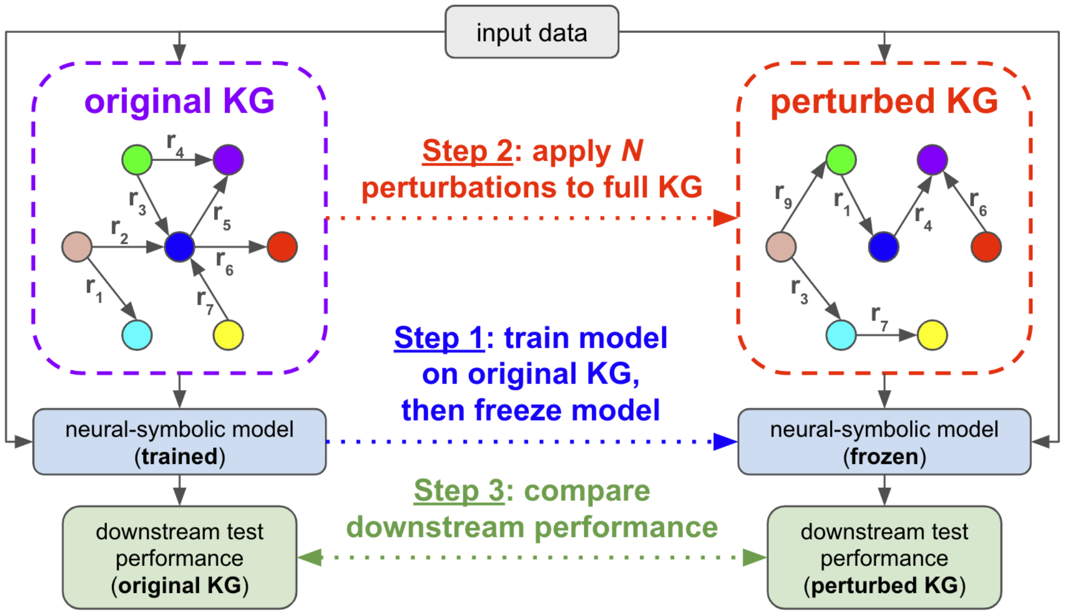
[28] Kaixin Ma, Jonathan Francis, Quanyang Lu, Eric Nyberg, and Alessandro Oltramari. Towards generalizable neuro-symbolic systems for commonsense question answering. *arXiv preprint arXiv:1910.14087*, 2019.

[29] Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, et al. Improving natural language inference using external knowledge in the science questions domain. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7208–7215, 2019.

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

[31] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[32] Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology*, 3(3):57:1–57:22, 2012.



input data
original KG
perturbed KG
Step 2: apply N perturbations to full KG
Step 1: train model on original KG, then freeze model
Step 3: compare downstream performance
neural-symbolic model (trained)
neural-symbolic model (frozen)
downstream test performance (original KG)
downstream test performance (perturbed KG)
r1
r2
r3
r4
r5
r6
r7
r9

Figure 2: **Proposed KG Perturbation Framework.** Our procedure consists of three main steps: **(1)** train the NSKG model on the original KG, then freeze the model; **(2)** obtain the perturbed KG by applying $N = |\mathcal{T}|$ perturbations to the full original KG; and (3) compare the perturbed KG's downstream performance to that of the original KG. If the perturbed KG's performance is not much worse than the original KG's performance, then this suggests that the model is not faithfully using the KG.

A Appendix

A.1 Diagram of KG Perturbation Framework

Figure 2 provides a big-picture schematic of our KG perturbation framework. For details about our high-level procedure, refer to the "High-Level Procedure" paragraph of Section 2.

A.2 Diagram of KG Perturbation Methods

Figure 3 illustrates our proposed KG perturbation methods. For explanation of our RL-based perturbation method (RL-RR), refer to Section 3. For descriptions of each heuristic-based perturbation method (RS, RR, ER, ED), refer to the "Baselines" paragraph of Section 4.

A.3 Edge Scoring Function ($s_{\mathcal{G}}$)

The choice of edge scoring function $s_{\mathcal{G}}$ is task-specific, since KGs from different tasks may differ greatly with respect to semantics or connectivity. For commonsense QA, we use the scoring function from [22]. For item recommendation, we use the scoring function from [23].

A.4 Additional KG Similarity Metrics

In order to measure how much the perturbed KG has deviated from the original KG, we need metrics that capture both semantic and structural similarity. Although ATS can reasonably quantify semantic differences between KGs, it is not sensitive to differences in connectivity structure. For instance, out of our five proposed perturbation methods, we can intuitively tell that ED destroys the most KG information, yet ED's ATS scores are consistently the best. To address this shortcoming of ATS, we consider two complementary metrics that do capture KG connectivity: SC2D and SD2. Later, in Appendix A.5.3, we validate ATS, SC2D, and SD2 by measuring how well they correlate with human judgment of KG readability and usability.

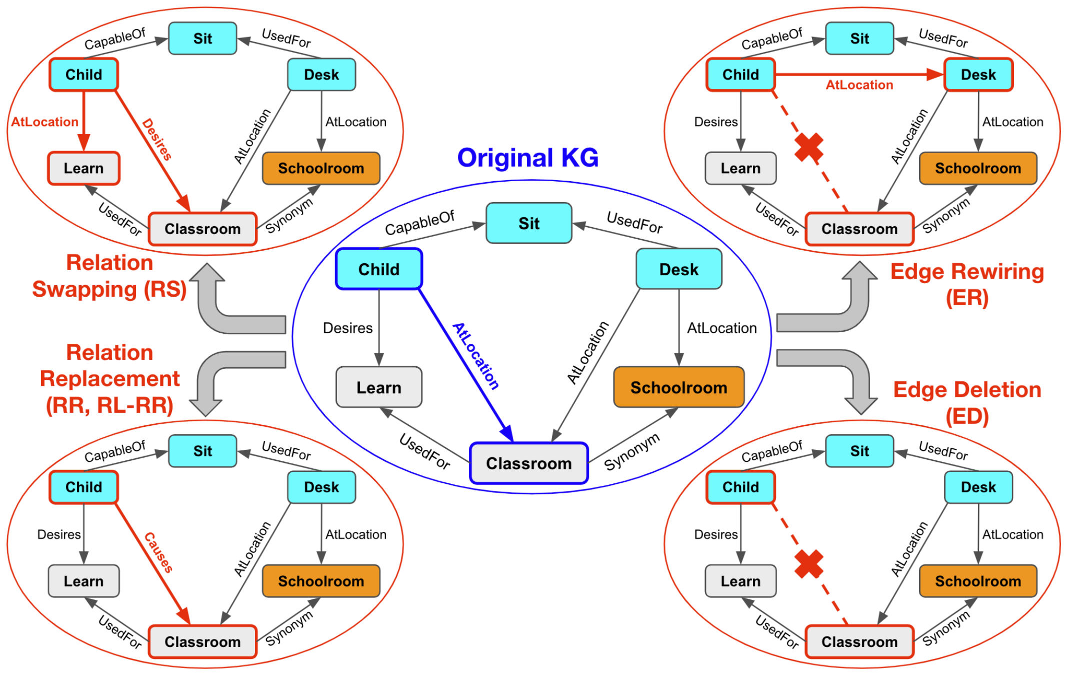


Figure 3: **Proposed KG Perturbation Methods.** We propose four heuristic-based perturbation methods and one RL-based perturbation method. In this diagram, we consider example edge (`Child, AtLocation, Classroom`) within a subgraph of the original ConceptNet KG (shown in blue). We illustrate how this edge (and possibly other edges) changes in response to different perturbation methods (shown in red). Unlike the heuristic-based methods (RS, RR, ER, ED), the RL-based method (RL-RR) is trained to maximize downstream performance and minimize original-perturbed KG semantic similarity.

Similarity on Clustering Coefficient Distribution (SC2D) SC2D measures structural similarity between two KGs and is derived from the local clustering coefficient [24, 25, 26]. For a given entity in $\mathcal{G}$ (treated here as undirected), the local clustering coefficient is the fraction of possible triangles through the entity that exist (i.e., how tightly the entity's neighbors cluster around it). For entity $e_i \in \mathcal{E}$, the local clustering coefficient is defined as $c_i = 2\text{Tri}(e_i)/(\deg(e_i)(\deg(e_i)-1))$, where $\text{Tri}(e_i)$ is the number of triangles through e_i, and $\deg(e_i)$ is the degree of e_i. For each relation $r \in \mathcal{R}$, let $\mathcal{G}^r$ be the subgraph of $\mathcal{G}$ consisting of all edges in $\mathcal{T}$ with r . That is, $\mathcal{G}^r = (\mathcal{E}, r, \mathcal{T}^{'})$, where $\mathcal{T}^{'} = \{(e, r, e') \mid e, e' \in \mathcal{E}\}$. Let $\mathbf{c}^r$ denote the $|\mathcal{E}|$-dimensional clustering coefficient vector for $\mathcal{G}^r$, where the ith element of $\mathbf{c}^r$ is c_i. Then, the mean clustering coefficient vectors for $\mathcal{G}$ and $\mathcal{G}'$ are $\mathbf{c}_o = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} \mathbf{c}^r$ and $\mathbf{c}_p = \frac{1}{|\mathcal{R}'|}\sum_{r\in\mathcal{R}'} \mathbf{c}^r$, respectively. Finally, SC2D is defined as $f_{\text{SC2D}}(\mathcal{G}, \mathcal{G}^{'}) = \frac{1}{\|\mathbf{c}_o - \mathbf{c}_p\|_2 + \epsilon}$, where ϵ is a small constant to avoid division by zero.

Similarity on Degree Distribution (SD2) SD2 measures structural similarity between two KGs, while addressing SC2D's ineffectiveness when the KGs' entities have tiny local clustering coefficients (e.g., the item KG used by recommender systems is roughly bipartite). In such cases, SC2D always equals roughly zero regardless of perturbation method, thus rendering SC2D useless. Let $\mathbf{d}^r$ denote the $|\mathcal{E}|$-dimensional degree vector for $\mathcal{G}^r$, where the ith element of $\mathbf{d}^r$ is $\deg(e_i)$. Then, the mean degree vectors for $\mathcal{G}$ and $\mathcal{G}'$ are $\mathbf{d}_o = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} \mathbf{d}^r$ and $\mathbf{d}_p = \frac{1}{|\mathcal{R}'|}\sum_{r\in\mathcal{R}'} \mathbf{d}^r$, respectively. Finally, SD2 is defined as $f_{\text{SD2}}(\mathcal{G}, \mathcal{G}^{'}) = \frac{1}{\|\mathbf{d}_o - \mathbf{d}_p\|_2 + \epsilon}$.

A.5 Additional Experiments

A.5.1 Commonsense QA

Here, we present more comprehensive experiments for commonsense QA. For convenience, we present a self-contained, more detailed description of the experiment settings below.

For commonsense QA, the NSKG models we experiment with are RN (with attentional path aggregation) [1, 12] and MHGRN [2], which have been shown to outperform non-KG models [19, 27] and a number of NSKG models [1, 28, 29, 30] on this task. For both models, we use a BERT-Base [19] text encoder. We evaluate on the CommonsenseQA (CSQA) [4] and OpenBookQA (OBQA) [31] datasets, using ConceptNet [20] as the KG. Performance is measured using accuracy, which is the standard metric for commonsense QA [1, 2]. We report performance and KG similarity for when all $|\mathcal{T}|$ edges in the KG are perturbed, averaged over three runs.

CSQA Results for CSQA are given in Table 2. For RN and MHGRN, we see that RL-RR achieves slightly worse accuracy than Original KG, while RS, RR, and ER perform on par with No KG. For both models, ED performs noticeably worse than No KG. We also observe that perturbed KGs produced by RS and ED have high ATS (i.e., semantic similarity to original KG), while RR, ER, and RL-RR achieve relatively low ATS.

OBQA Results for OBQA are shown in Table 3. For RN, we see that RL-RR actually obtains better accuracy than Original KG. For MHGRN, RL-RR yields marginally worse accuracy than Original KG. Meanwhile, for both RN and MHGRN, all heuristics uniformly achieve similar accuracy as Original KG, which itself significantly outperforms No KG.

Analysis Tables 2-3 demonstrate that perturbing a KG does not necessarily imply decreased performance, nor does it guarantee the creation of invalid or novel facts. As shown by the KG similarity scores, some perturbation methods cause greater semantic or structural KG changes than others. While ATS varies considerably across perturbation methods, SC2D and SD2 are quite low for all perturbation methods, indicating consistently low structural similarity between original and perturbed KG. RL-RR and RR collectively have the lowest SC2D and SD2 for CSQA, while RL-RR has the lowest SC2D and SD2 for OBQA. Notably, across all perturbation methods and models, RL-RR attains the highest accuracy while also having the lowest KG similarity scores overall. The results of a T-test (three runs for both models) show that RL-RR achieves a statistically significant improvement over its heuristic counterpart, RR. Still, even RR has a fairly high accuracy to KG similarity ratio. This suggests that our NSKG models are not using the KG in a faithful way, since RL-RR and RR can both achieve high performance despite extensively corrupting the original KG's semantic and structural information.

A.5.2 Item Recommendation

Here, we present more comprehensive experiments for item recommendation. For convenience, we present a self-contained, more detailed description of the experiment settings below.

The NSKG recommender systems we consider are KGCN [7] and RippleNet [5]. We evaluate these models on the Last.FM [32] and MovieLens-20M [21] datasets, using the item KG from [6]. As mentioned in Section 1, item KGs have been shown to benefit recommender systems in cold start scenarios [5]. Therefore, following [5], we simulate a cold start scenario by using only 20% and 40% of the train set for Last.FM and Movie Lens-20M, respectively. Performance is measured using AUC, which is the standard metric for item recommendation [7, 5]. Since the item KG is almost bipartite, SC2D is not meaningful here, since the local clustering coefficient of each item in the KG is extremely small (Appendix A.4). Thus, for item recommendation, we do not report SC2D. We report performance and KG similarity for when all $|\mathcal{T}|$ edges in the KG are perturbed, averaged over three runs.

MovieLens-20M Results for MovieLens-20M are displayed in Table 4. For both KGCN and RippleNet, we find that relation-based perturbation methods tend to perform on par with Original KG. Here, ER is the better of the two edge-based perturbation methods, performing about the same as Original KG for KGCN, but noticeably worse for RippleNet. Somehow, for both KGCN and RippleNet, ED achieves even worse AUC than No KG. On the other hand, we see that ED achieves very high ATS, while RS, RR, ER, and RL-RR achieve more modest ATS scores.

Last.FM Results for Last.FM are displayed in Table 5. For both KGCN and RippleNet, we see that RS, RR, and RL-RR achieve about the same AUC as Original KG, with RL-RR slightly outperforming Original KG. Furthermore, ER performs similarly to Original KG for KGCN, but considerably worse for RippleNet. ED's AUC is on par with No KG's for KGCN and much lower than No KG's for RippleNet. We also observe that the perturbed KGs produced by ED have high ATS, while RS, RR, ER, and RL-RR achieve more modest ATS scores.

Table 2: **Comparison of Perturbation Methods on CSQA.** We follow the standard protocol of reporting test accuracy on the in-house data split from [1], as the official test labels are not available. The results here are from perturbing all facts in the original KG and are averaged over three runs.

	CSQA							
	RN				**MHGRN**			
Method	Acc (↑)	ATS (↓)	SC2D (↓)	SD2 (↓)	Acc (↑)	ATS (↓)	SC2D (↓)	SD2 (↓)
No KG	53.41	-	-	-	53.41	-	-	-
Original KG	56.87	0.940	-	-	57.21	0.940	-	-
Relation Swapping (RS)	53.42	0.831	0.168	6.20E-3	53.42	0.831	0.168	6.20E-3
Relation Replacement (RR)	53.42	0.329	**0.100**	1.70E-3	52.22	0.329	**0.100**	**1.70E-3**
Edge Rewiring (ER)	53.42	0.505	0.131	2.31E-3	52.22	0.505	0.131	2.31E-3
Edge Deletion (ED)	52.21	0.933	0.144	2.00E-3	51.00	0.933	0.144	2.00E-3
RL-RR	**55.21**	**0.322**	0.102	**1.66E-3**	**55.52**	**0.314**	0.101	1.78E-3

Table 3: **Comparison of Perturbation Methods on OBQA.** We report test accuracy on OBQA using the official data split. The results here are from perturbing all facts in the original KG and are averaged over three runs.

	OBQA							
	RN				**MHGRN**			
Method	Acc (↑)	ATS (↓)	SC2D (↓)	SD2 (↓)	Acc (↑)	ATS (↓)	SC2D (↓)	SD2 (↓)
No KG	62.00	-	-	-	62.00	-	-	-
Original KG	66.80	0.934	-	-	68.00	0.934	-	-
Relation Swapping (RS)	67.00	0.857	0.189	7.79E-3	67.30	0.857	0.189	7.79E-3
Relation Replacement (RR)	66.80	0.269	**0.105**	1.84E-3	67.60	0.269	0.105	1.84E-3
Edge Rewiring (ER)	66.60	0.620	0.172	7.31E-3	67.00	0.620	0.172	7.31E-3
Edge Deletion (ED)	66.80	0.923	0.155	2.19E-3	67.60	0.923	0.155	2.19E-3
RL-RR	**67.30**	**0.255**	0.108	**1.79E-4**	**67.70**	**0.248**	**0.104**	**1.75E-4**

Analysis Like in commonsense QA, Tables 4-5 shows that NSKG models can perform well even when the KG has been drastically perturbed. Using the T-test with three runs, for almost all perturbation methods, we find a statistically insignificant difference between the perturbed KG's AUC and the original KG's AUC. Whereas ATS again varies greatly across perturbation methods, all perturbation methods have fairly low SD2 (except RR on Last.FM), indicating that structural similarity between original and perturbed KG is consistently low. In particular, across both datasets and models, RL-RR has the highest AUC overall, while also having the lowest KG similarity scores overall. This serves as additional evidence that the model is not using the KG faithfully, since RL-RR achieves high performance despite significantly perturbing the original KG's semantic and structural information.

A.5.3 Auxiliary Experiments and Analysis

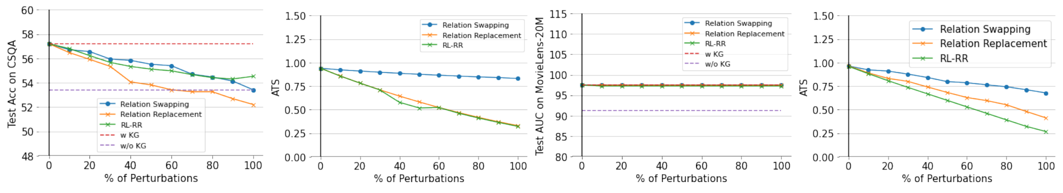
Varying Perturbation Level For a subset of model-dataset-perturbation settings, we measure the performance and ATS of various perturbation methods as a function of the percentage of KG edges perturbed. For MHGRN on CSQA, Figure 4a shows that, across all levels of perturbation, RL-RR maintains higher accuracy than No KG. Meanwhile, RS's accuracy reaches No KG's accuracy at 100% perturbation, and RR's does so at 60% perturbation. In Figure 4b, we see that RL-RR's and RR's ATS drop significantly as the perturbation percentage increases, whereas RS's ATS remains quite high even at 100% perturbation. For RippleNet on MovieLens-20M, Figure 4c shows a flat performance curve for all perturbation methods. Meanwhile, for all perturbation methods in Figure 4d, ATS decreases steadily as the number of perturbations increases, with RL-RR's ATS dropping most. These findings support the hypothesis that KG perturbation does not imply performance decrease or KG corruption. Building on the results of previous experiments, in both model-dataset settings, RL-RR largely maintains the model's performance despite also heavily perturbing the KG's semantics. Interestingly, for RippleNet on MovieLens-20M, performance is completely unaffected by KG perturbation, even though the KG's semantic information is apparently being corrupted.

Table 4: **Comparison of Perturbation Methods on MovieLens-20M.** We report the test AUC on MovieLens-20M using the data split of [5]. The results here are from perturbing all facts in the original KG and are averaged over three runs.

	MovieLens-20M					
	KGCN			**RippleNet**		
Method	Acc (↑)	ATS (↓)	SD2 (↓)	Acc (↑)	ATS (↓)	SD2 (↓)
No KG	91.30	-	-	91.30	-	-
Original KG	96.62	0.960	-	97.46	0.960	-
Relation Swapping (RS)	**96.62**	0.678	6.14E-4	**97.46**	0.678	6.14E-4
Relation Replacement (RR)	96.50	0.413	**7.74E-5**	97.45	0.413	**7.74E-5**
Edge Rewiring (ER)	96.24	0.739	3.16E-4	93.42	0.739	3.16E-4
Edge Deletion (ED)	90.36	0.982	1.02E-4	90.22	0.982	1.02E-4
RL-RR	96.53	**0.401**	2.23E-4	97.25	**0.268**	2.21E-4

Table 5: **Comparison of Perturbation Methods on Last.FM.** We report the test AUC on Last.FM using the data split of [5]. The results here are from perturbing all facts in the original KG and are averaged over three runs.

	Last.FM					
	KGCN			**RippleNet**		
Method	Acc (↑)	ATS (↓)	SD2 (↓)	Acc (↑)	ATS (↓)	SD2 (↓)
No KG	50.75	-	-	50.75	-	-
Original KG	55.99	0.972	-	56.23	0.972	-
Relation Swapping (RS)	55.98	0.681	3.05E-2	56.23	0.681	3.05E-2
Relation Replacement (RR)	55.98	0.415	0.339	56.22	0.415	0.339
Edge Rewiring (ER)	55.98	0.320	6.28E-3	53.74	0.320	6.28E-3
Edge Deletion (ED)	50.96	0.941	3.38E-3	45.98	0.941	3.38E-3
RL-RR	**56.04**	**0.320**	**1.30E-3**	**56.28**	**0.310**	**1.20E-3**



(a) CSQA Accuracy (b) CSQA ATS (c) MovieLens-20M AUC (d) MovieLens-20M ATS

Figure 4: **Varying Perturbation Level.** Performance and ATS with respect to perturbation level, for MHGRN on CSQA and RippleNet on MovieLens-20M. The horizontal axis denotes the percentage of perturbed edges in the KG.

Noisy Baselines To see if KGs produced by our perturbation methods are capturing more than just random noise, we compare them to several noisy baselines. Table 6 contains results for three noisy baselines on commonsense QA: (1) replace subgraph embedding with zero vector, (2) replace subgraph embedding with random vector, and (3) replace entity/relation embeddings with random vectors. For CSQA, the noisy baselines perform noticeably worse than both Original KG and RL-RR, while being on par with No KG (Table 2). For OBQA, the noisy baselines' performance is slightly better than No KG, but considerably worse than not only Original KG, but all of the perturbation methods (Table 3). Table 7 displays results for our noisy baseline in item recommendation, which entails randomizing each entity's neighborhood. We find that KGCN performs about the same for this noisy baseline as for Original KG and our best perturbation methods, whereas RippleNet performs much worse (Tables 4-5). RippleNet may be more sensitive than KGCN to entity neighbor randomization because RippleNet considers directed edges. This is supported by RippleNet's performance dropping when we perturb edge connections (Tables 4-5). In both tasks, the noisy baselines show that our perturbation methods yield KGs that capture measurably useful information

Method	CSQA		OBQA	
	RN	MHGRN	RN	MHGRN
No KG	53.41	53.41	62.00	62.00
Original KG	56.87	57.21	66.80	68.00
Zero Subgraph Emb.	53.10	53.99	64.80	66.40
Rand. Subgraph Emb.	52.60	52.48	64.75	65.90
Rand. Ent./Rel. Emb.	53.02	54.03	64.45	64.85

Table 6: **Noisy Baselines for Commonsense QA.** Test accuracy of noisy baselines on CSQA and OBQA.

Method	MovieLens-20M		Last.FM	
	KGCN	RippleNet	KGCN	RippleNet
No KG	91.30	91.30	50.75	50.75
Original KG	96.62	97.46	55.99	56.23
Rand. Ngbd.	96.21	92.11	55.91	51.04

Table 7: **Noisy Baseline for Item Recommendation.** Test AUC of noisy baseline on MovieLens-20M and Last.FM.

beyond just noise. For KGCN, the unexpected discovery that noisy baselines perform similarly to Original KG suggest that even noisy KGs can contain useful information for NSKG models.

Human Evaluation of KG Explanations We conduct a user study to measure KGs' ability to provide plausible explanations. Here, we consider the original KG and the RL-RR KG. For both KGs, we sample 30 questions from the CSQA and OBQA test sets which were correctly answered by MHGRN. For each question, we retrieve the top-scoring path for each answer choice via MHGRN's path decoder attention. We then ask three human annotators to rate each path for readability and usability, with ratings aggregated via majority voting. Readability ($[0, 1]$ scale) is whether the path makes sense. Usability ($[0, 1, 2]$ scale) is whether the path is relevant to the given question-answer pair. We obtain a Fleiss' κ of 0.1891, indicating slight inter-annotator agreement. In Table 8, we see that Original KG and RL-RR scored very similarly, except for RL-RR being much worse on OBQA. This shows that RL-RR can maintain model performance while corrupting the KG semantically and structurally. Nonetheless, Original KG and RL-RR both got relatively low ratings overall, indicating the implausibility of NSKG models' explanations.

Method	CSQA		OBQA	
	Read	Use	Read	Use
Orig. KG	0.081	0.720	0.111	0.714
RL-RR	0.115	0.706	0.111	0.397

Table 8: **Human Evaluation.** Human ratings of the readability (Read) and usability (Use) of top-scoring KG paths from MHGRN.

OBQA's much larger usability gap between Original KG and RL-RR can be explained by the fact that CSQA is constructed from ConceptNet. Every CSQA question-answer is based on ConceptNet entities/relations, so a random ConceptNet subgraph is more likely to have semantic overlap with a CSQA question-answer than with an OBQA question-answer. Hence, a perturbed ConceptNet subgraph may also be more likely to overlap with a CSQA question-answer, and so perturbing the KG might have a smaller impact on human judgments of CSQA path usability. This does not say anything about the model's performance on CSQA and OBQA, as high model performance does not necessarily imply high explanation quality. In fact, our user study disproves this connection.

Validation of KG Similarity Metrics Using the results of our human evaluation, we validate our three proposed KG similarity metrics: ATS, SC2D and SD2. We find that the Pearson correlation coefficient between the human evaluation scores in Table 8 and the three KG similarity scores in Tables 2-3 are 0.845, 0.932 and 0.932, respectively. This indicates high correlation and demonstrates that the KG similarity metrics aptly capture how well a perturbed KG preserves semantic and structural information from its original KG.

Why do perturbed KGs sometimes perform better than the original KG? In our experiments, relation-based perturbations (RS, RR, RL-RR) generally outperform edge-based perturbations (ER, ED). Also, we have found that the original KG can contain noisy relation annotations which are sometimes "corrected" by relation-based perturbations. In certain cases, this may result in the perturbed KG achieving slightly higher performance than the original KG (RR and RL-RR for RN-CSQA; RL-RR for Last.FM). Similarly, in our user study, despite all questions being correctly answered by the model, there were some RL-RR explanations that received higher readability/usability ratings than their original KG counterparts. Although the original KG achieved higher human ratings than the RL-RR KG did overall, both KGs still achieved relatively low ratings with respect to our scales. While our main argument centers on NSKG models' flaws, this counterintuitive finding suggests that KGs themselves are flawed too, but in a way that can be systematically corrected.