LEARNING TO DECEIVE KNOWLEDGE GRAPH AUGMENTED MODELS VIA TARGETED PERTURBATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Symbolic knowledge such as entities and relational triples stored in knowledge graphs has increasingly gained popularity in neural-symbolic architectures applied to machine learning tasks (e.g., question answering, recommender systems). In addition to yielding improved performance on downstream tasks, these symbolic structures in conjunction with their associated (attention) weights are often used as explanation for the model's prediction, providing "insights" to practitioners. In this paper, we question the faithfulness of these symbolic explanations, and demonstrate that by a learned perturbation strategy, or sometimes a simple heuristic, one can produce deceptive symbolic structures which significantly deviate from the original semantics. Our method learns a reinforcement learning policy to identify alternative relations to replace the original ones in the knowledge graph such that the downstream accuracy is maximally preserved. Across multiple models and tasks, our approach changes the structures entirely while resulting in only marginal or no drop in performance. Our results raise doubt about the faithfulness of the learned structures as model explanation and the reliability of current neural-symbolic models in leveraging symbolic knowledge. [1]

1 INTRODUCTION

Recently the neural-symbolic models which incorporate symbolic structures from external knowledge graphs (KGs) are gaining growing popularity in machine learning tasks such as recommender systems (Wang et al., 2018b; 2019a;b), question answering (Lin et al., 2019; Feng et al., 2020; Lv et al., 2020), and natural language inference (Kapanipathi et al.; Wang et al., 2019c). A typical KG-augmented (neural-symbolic) model either leverages the encoding of the grounded relational structures as additional input features (Lin et al., 2019; Feng et al., 2020), or models the relational connections in KG to propagate information among the entities (*i.e.*, message passing) (Wang et al., 2018b; 2019a). By doing so, these methods successfully improve the performance on downstream tasks by bringing symbolic knowledge which is essential yet absent in the input as in commonsense reasoning tasks (Talmor et al., 2018), or help to address data sparsity and cold start issues as in recommender systems (Wang et al., 2018a). In addition to enhancing model performance, such symbolic knowledge, together with its learned (attention) weights, are used as explanations for the model predictions (Feng et al., 2020; Lin et al., 2019; Cao et al., 2019; Gao et al., 2019), providing "insights" to practitioners for understanding the reasoning process.

Due to the increased usage of neuro-symbolic models for various tasks, it is of paramount importance to analyze the structures learned by these models. In this paper, we question whether the KG structures associated with the data instances provide faithful explanation for the model's predictions and information carried by these structures–*i.e.*, relations and connections between entities–necessarily contribute to the performance improvement on the models. Through a series of experiments on diverse tasks and KG-augmented models, we show that one can manipulate the KG structures without significant impact on the performance of the model, yielding deceptive model explanation and raising doubts on what information from the KG are actually captured. Our pilot study applies simple heuristics to perturb all the edges and relations in the KG. In some cases, despite a deceptive symbolic structure, which completely deviates from the original KG, we observe that existing models still manage to maintain their performance on downstream tasks. For cases where model

[1] Code has been uploaded in the supplementary materials and will be published.

performance has noticeable drops, our human evaluation on the model explanation shows that the perturbed structure does not provide faithful explanations.

The preliminary results indicate that KG can be easily manipulated and lost its benefit. It brings more worrisome scenario that KG is perturbed in a more systematic way so that even advanced KG-augmented models can be deceived by such a perturbed KG without noticeable performance drop. To show the feasibility of such perturbation policies, we conduct a more sophisticated study where we propose a reinforcement learning (RL) based perturbation algorithm. We leverage the Deep Q-network (DQN) to model the change of downstream task performance given specific perturbation actions, and restrict the agent's action space to ensure nearly maximal perturbation. In this way, the agent learns a perturbation policy with a dual objective of maximally perturbing the graph but maintaining the original accuracy. We show that the agent learns to identify which edges or relations in the KG to perturb while preserving the competitive performance of these KG-augmented models. Unlike the heuristic methods, the RL agent can be tailored to a certain KG-augmented model by making use of its performance change as the training signal, and thus yield a perturbed graph with less structural and semantic information and a better performance compared with heuristic methods.

In summary, our main findings and contributions are as follows: (1) **Knowledge Graph Perturbation Framework:** We study perturbations to the knowledge graph along two dimensions, *semantically* by randomization of the relation types while holding the distribution of those types to be the same (and therefore controlling for the different factors), and *structure-based perturbations* that changes the connections in the knowledge graph. (2) **RL-based Targeted Perturbation:** We demonstrate that on some tasks it is non-trivial to achieve our goal of perturbing a KG while maintaining downstream performance with simple perturbations. Therefore we present RL-based perturbation policy learning to maximally perturb the graph while trying to maintain the downstream performance. (3) We find via our experiments that even with a KG that is corrupted by a sizeable amount, we are able to maintain the downstream performance on most tasks. Through our different types of perturbations and human evaluation, we demonstrate how faithful are the explanations generated by KG-augmented models and how much are the models sensitive to changes in the KG.

2 Deceiving KG-Augmented Models by Graph Perturbation

We aim to investigate whether KGs are being leveraged appropriately by the KG-augmented models. A KG is denoted as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ and $\mathcal{R}$ are the entity set and relation set respectively. It consists of a set of relational edges $\mathcal{T} = \{(h, r, t) | h, t \in \mathcal{E}, r \in \mathcal{R}\}$. Let $\mathcal{N}_L(e)$ denote the set of L-hop neighbors for the entity $e \in \mathcal{E}$. Let $\mathcal{G}^{'} = (\mathcal{E}, \mathcal{R}, \mathcal{T}^{'}), \mathcal{T}^{'} \neq \mathcal{T}$ be a perturbed version of $\mathcal{G}$. We define a KG evaluation metric function $f(\mathcal{G}, \mathcal{G}^{'})$ which quantifies how much relational information in the KG is *preserved* after perturbation. We further define a performance measurement function $g(\mathcal{G}, \mathcal{G}^{'})$ which quantifies the difference in a model's performance, *e.g.*, accuracy on the downstream task. Our goal (*i.e.*, deceiving KG-augmented models) is to identify a *perturbed* KG, $\mathcal{G}^{'}$, which maximally deviates from the original KG, $\mathcal{G}$; while minimally affecting the downstream performance of the KG-augmented model. This perturbation can be formalized into the following objective.

$$\mathcal{G}^{*} = \arg\min_{\mathcal{G}'} f(\mathcal{G}, \mathcal{G}^{'}) + g(\mathcal{G}, \mathcal{G}^{'}). \tag{1}$$

In specific, the KG-augmented model is well-trained on $\mathcal{G}$ to get the original performance and fixed. Later the model is tested on $\mathcal{G}^{'}$ to get its performance after perturbation. In this work, we look into two different tasks across different domains, *e.g.*, commonsense question answering (QA) and recommender system, as introduced in the rest of this section.

Commonsense Question Answering. Given a question q and a set of choices $\mathcal{C}$, the task requires a system to select one of the choices $a \in \mathcal{A}$ as the answer. To tackle the task, the KG-augmented QA systems such as KagNet (Lin et al., 2019) and MHGRN (Feng et al., 2020) first extract the set of entities $\mathcal{E}_{(q,a)} \subset \mathcal{E}$ mentioned in the pair of (q, a) with existing entity recognition tools and later construct a sub-graph $\mathcal{G}_{(q,a)}$ containing $\mathcal{E}_{(q,a)}$ and their multi-hop relations in the KG. Then the system encodes the (q, a) pair to a context embedding $\mathbf{c}$ and encodes $\mathcal{G}_{(q,a)}$ to a knowledge embedding $\mathbf{k}$. Finally, both $\mathbf{c}$ and $\mathbf{k}$ are concatenated for calculating the plausibility score. The choice with the highest score is selected as the answer. Two types of graph encoders are generally adopted to encode $\mathcal{G}_{(q,a)}$ to $\mathbf{k}$ as described below.

(1) Graph Neural Networks (Kipf & Welling, 2016; Schlichtkrull et al., 2018) firstly represent each node by its feature embedding $\{\mathbf{h}_i^0\}, i \in \mathcal{G}_{(q,a)}$ and then iteratively update a node's representation by aggregating its neighbors' embedding, i.e., $\mathbf{h}_i^l = \text{Aggregation}(\{\mathbf{h}_j^{l-1} | j \in \mathcal{N}_1(i)\})$. Finally, a pooling over all the node embeddings is used as the graph representation: $\mathbf{k} = \text{Pooling}(\{\mathbf{h}_i^L\})$ with L indicating each node receives information from its neighbors up to L hops.

(2) Path-Based Models (Lin et al., 2019; Santoro et al., 2017) treat an input graph as a set of relational paths $\mathcal{P}$ connecting the entities. Then each path $p_i \in \mathcal{P}$ is encoded as an embedding $\mathbf{p}_i$ and further aggregated to obtain the knowledge embedding: $\mathbf{k} = \text{Aggregation}(\{\mathbf{p}_i\})$.

In both graph encoders, the subgraph $\mathcal{G}_{(q,a)}$ together with the aggregation weights serve as the explanation for the prediction.

Recommender Systems. In this task, we consider a set of users $\mathcal{U} = \{u_1, u_2, ..., u_M\}$ and a set of items $\mathcal{V} = \{v_1, v_2, ..., v_N\}$, as well as their interaction matrix $\mathbf{Y} \in \mathcal{R}^{M \times N}$. An observed interaction ($y_{uv} = 1$) indicates that the user u engages with the item v before while the task is to predict the unobserved interaction ($y_{uv} = 0$). The KG-augmented recommender systems assume there exists an item KG which provides bountiful relational connections between the items. They further represent each user and item as embeddings $\mathbf{u}$ and $\mathbf{v}$ and then use their interaction, e.g., inner product for recommending v to u. Below we introduce two recent works we investigate in this paper which differ in how they leverage the item KG and represent the users and items.

KGCN (Wang et al., 2019b) represents an item as each aggregation over its neighbors in the KG, i.e., $\mathbf{v} = \text{Aggregation}_u(\{\mathbf{v}^{'} | v^{'} \in \mathcal{N}_1(v)\})$, where the aggregation weights are personalized for u. This aggregation is repeated L times to obtain information from L-hops neighbors (similar to GNN). The user embedding is initialized randomly and learned during training. In contrast, RippleNet (Wang et al., 2018b) represents each user u as the aggregation over the L-hops neighbors of the items $\mathcal{V}_u$ which u has been interacted with before, i.e., $\mathbf{u} = \text{Aggregation}(\{\mathbf{v} | v \in \mathcal{N}_L(v^{'}), \forall v^{'} \in \mathcal{V}_u\})$. The item embedding is initialized randomly and learned during training. In both KGCN and RippleNet, the aggregation weights on the items' neighbors are used to explain the recommendation results.

In the rest of the paper, we will first present the pilot study that motivates the work, and then describe the targeted RL perturbation policy that we propose for our analysis. Finally we detail our experimental setup and show our results and analysis that support the claim of the paper.

3 Pilot Study with Simple Perturbation Heuristics

As a pilot study to motivate our work, we first consider two simple strategies to perturb the KG: (1) **Relation Swapping (RS)**: We randomly choose two triples (edges) in the KG and swap their relations. This heuristic changes the semantic of the triples while leaving the underlying connectivity structure unchanged. (2) **Relation Replacement (RR)**: We randomly choose a triple (h, r_1, t) and then replace its relation with another one r_2 that yields the most invalid triples, i.e., $r_2 = \text{argmin}_{r \in \mathcal{R}} s_{\mathcal{G}}(h, r, t)$ where $s_{\mathcal{G}}(\cdot)$ is a KG scoring function trained on $\mathcal{G}$. This heuristic also does not perturb KG's structure but its semantic.

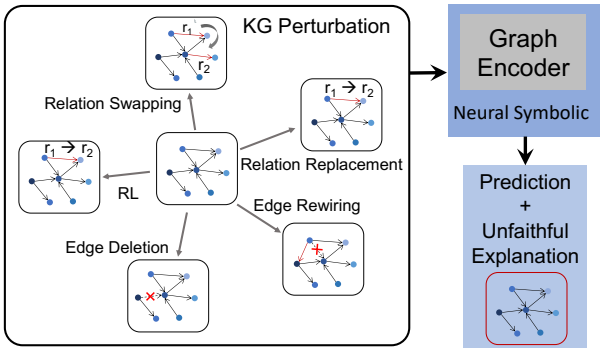
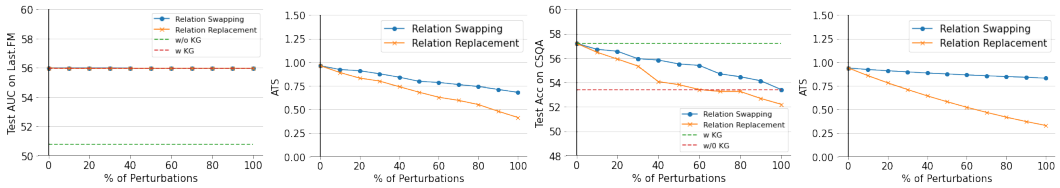


Figure 1: **Proposed KG perturbation framework.**

To quantify to what extent we perturb the KG, we propose an automatic measurement called **Aggregated Triple Score (ATS)**. It is defined as the average scores given by the KG scoring function mentioned above over all the triples present in the KG, i.e., $f^{ATS}(\mathcal{G}, \mathcal{G}^{'}) = \frac{1}{|\mathcal{T}^{'}|} \sum_{(h,r,t) \in \mathcal{T}^{'}} s_{\mathcal{G}}(h, r, t)$.

We conduct our pilot study on both of commonsense QA and recommendation system tasks. For commonsense QA, we adopt MHGRN (Feng et al., 2020), a hybrid of GNN and path-based model, on a commonsense benchmark dataset, CSQA (Talmor et al., 2018). For recommendation system, we adopt KGCN (Wang et al., 2019b) on the Last.FM (Rendle, 2012) dataset.



(a) Test AUC on Last.FM (b) ATS on Last.FM (c) Test Accuracy on CSQA (d) ATS on CSQA

Figure 2: Variation of Performance and Evaluation Metrics with scale of perturbation on KGCN with Last.FM and MHGRN with CSQA.

Observations and Discussion. We show curves of how the performance and the ATS metric change across different percentages of relation-based perturbations in Figure 2. We observe that simple heuristics are able to perturb the KG to a large extent (Figure 2-b) while maintaining the performance in the recommendation task (Figure 2-a). This casts doubt on the faithfulness of the symbolic knowledge for KGCN and motivates us to investigate other neural-symbolic models as well.

On the contrary, for MHGRN with CSQA we observe that indeed randomization of relations of the KG leads to a clear drop in performance (test accuracy), as shown in Figure 2-c. To further examine whether random noise could yield similar performance drop as simple heuristics do, we design two controlled noisy baselines where we replace the knowledge embeddings from MHGRN with random vectors and zero vectors respectively. By doing so, the model achieves 52.48% and 53.99% in accuracy, indicating that the simple heuristics just yield some meaningless knowledge embeddings. Since this is due to the fact that simple heuristics perturb KG blindly, we ask the question whether we can maintain the performance while corrupting a KG with some sophisticated perturbation policies. To this end in the next section we propose a RL based perturbation policy.

4 Policy Learning for Targeted KG Perturbation

In this section we introduce our targeted perturbation policy which learns to maximally perturb the graph while maintaining the performance on the downstream task. The policy only considers one type of perturbation which is replacing the relations. To achieve this goal, we adopt Deep Q-networks (DQN) (Mnih et al., 2015), a deep RL algorithm to learn the target policy. We model the perturbation procedure as a Finite Horizon Markov Decision Process (S, A, P, R, γ), where S is the state space, A is the action space, P is the transition probability, R is the reward function and γ is the discount factor. With this setting, DQN aims at learning an approximated Q function $Q(s_t, a_t)$ for state s_t and action a_t at step t. The detailed definitions are as follows.

States. The state space S consists of all the possible $\mathcal{G}$ that can be obtained via our perturbation. Since we only consider replacing relations, the state space is all the graphs with the same structure as the original graph.

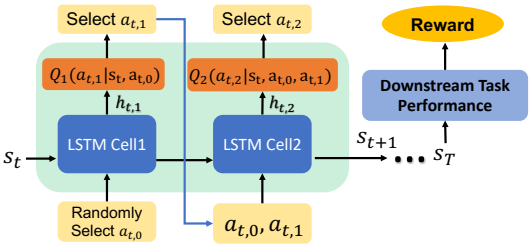


Figure 3: **The proposed DQN architecture for targeted KG perturbation.**

The DQN requires the embeddings s_t of the current state $\mathcal{G}_t$ as input. A common approach to represent the current state is by GNN-based graph encoder $s_t = \text{GNN}(\mathcal{G}_t)$. However, such method is not suitable for our task, since one of our goal is to examine whether these graph encoders have captured the information of KG. Thus it will conflict with our goal if we use our questioned object in our examining method. Therefore, we alternatively leverage a LSTM (Hochreiter & Schmidhuber, 1997) for the state embedding. At each step, we feed the embedding of the chosen action a_t into LSTM, and then the perturbed graph is represented by the output hidden vector of LSTM, *i.e.*, $h_t = \text{LSTM}(s_0, [a_1, a_2, ..., a_t])$. However, the perturbed actions $a_1, a_2, ..., a_t$ are fed sequentially to LSTM, and thus the state embedding s_t will not preserve the order-invariant property. To alleviate this issue, we randomly shuffle the order of actions and re-calculate the state embeddings during training of the DQN. Thus the model is forced to learn order invariant embeddings.

Actions. A simple action space for replacing relations can be defined as the set of all possible edges but it will make the action space very large and pose a huge scalability problem. Instead we break

down the relation replacement action into a triple of subactions, $a_t = (a_t^{(0)}, a_t^{(1)}, a_t^{(2)})$, where $a_t^{(0)}$ is to choose the head entity, $a_t^{(1)}$ is to choose a neighbour entity and finally $a_t^{(2)}$ is to choose a relation type that substitutes the relation between the chosen entities. We use a hierarchical system of two DQNs to choose these. $a_t^{(0)}$ is chosen randomly. First DQN takes the state and $a_t^{(0)}$ to sample $a_t^{(1)}$ and the second DQN follows similar procedure to sample $a_t^{(2)}$.

We further restrict the action space to the top K actions with the lowest ATS in the original action space for $a_t^{(2)}$. Thus we try to ensure a drastic drop in evaluation metrics with a reduced action space.

Reward. The reward is designed to reflect the change of downstream task performance. For QA tasks, we minimize the KL divergence between the predicted distribution and the distribution over right answers. The distribution over right answers is a dirac distribution over the right answer. We set the reward as the decrease in KL-Divergence from the previous state. The reward may be at very small scale, so we record the average of the absolute value of rewards and scale it to a pre-set value. For recommendation task, we directly use the development AUC metric as the reward.

DQN Architecture and Training Process. The proposed DQN architecture is illustrated in Figure 3. At the step t of perturbing triples in the KG, the state embedding s_t together with the chosen actions are fed into the LSTM Cell to form the Q functions Q_1 and Q_2 as follows:

$$Q_1(a_t^{(1)}|s_t, a_t^{(0)}) = \langle MLP(a_t^{(1)}), MLP(h_t^{(1)})\rangle; \tag{2}$$

$$Q_2(a_t^{(2)}|s_t, a_t^{(0)}, a_t^{(1)}) = \langle MLP(a_t^{(2)}), MLP(h_t^{(2)})\rangle; \tag{3}$$

where $h_t^{(1)} = \text{LSTMCell}_1(s_t, a_t^{(0)})$ and $h_t^{(2)} = \text{LSTMCell}_2(s_t, [a_t^{(1)}, a_t^{(0)}])$ are the immediate hidden representation of current state. Actions $a_t^{(1)}$ and $a_t^{(2)}$ are chosen via Q function ϵ-greedily during training and greedily during evaluation.

Evaluating the KG on downstream task is heavily time-consuming, and our task also requires a large number of perturbations. Therefore, we only get rewards every T steps. T can be as large as hundreds. In this way, the Q function $Q(s_t, a_t)$ is to predict the expectation of reward at step T if we choose action a_t at current state s_t.

The training of DQN follows the common approach. Q functions are updated through Q-iterations minimizing the Bellman Error as follows.

$$\theta \leftarrow \theta - \alpha \nabla_\theta (\gamma \max_{a_t^{(2)}} Q_{2,\phi'}(a_t^{(2)}|s_t, a_t^{(0)}, a_t^{(1)}) - Q_{1,\theta}(a_t^{(1)}|s_t, a_t^{(0)}))^2; \tag{4}$$

$$\phi \leftarrow \phi - \alpha \nabla_\phi ((r + \gamma \max_{a_{t+1}^{(1)}} Q_{1,\theta'}(a_{t+1}^{(1)}|s_{t+1}, a_{t+1}^{(0)})) - Q_{2,\phi}(a_t^{(2)}|s_t))^2. \tag{5}$$

In this way, we are able to train the DQN and learn our targeted perturbation policy.

5 Experiments

In this section, we evaluate the effectiveness of our heuristic and RL-based perturbation in perturbing KG as well as preserving performance of a fixed model on commonsense QA and recommender system tasks respectively. We denote our RL based perturbation policy by **RL-Relation Replacement(RL-RR)**. We also conduct human evaluation on the explanation given by the symbolic knowledge from both unperturbed and perturbed KGs to cast doubt on their faithfulness.

5.1 Baselines for Graph Perturbation

In addition to the Relation Swapping(RS) and Relation Replacement(RR) perturbation heuristics (§ 3), we further introduce two other perturbation methods as baselines in our experiments.

Edge Rewiring (ER): We first randomly select an edge (h, r, t) in the KG. Let $k \in \mathcal{E} \backslash \mathcal{N}_1(h)$. Then we rewire the edge to another one with the same relation, i.e., $\mathcal{T} \leftarrow (\mathcal{T} \setminus \{(h, r, t)\}) \cup \{(h, r, k)\}$. It alters the structure of the graph but keep the relation distribution across the KG constant.

Edge Deletion (ED): We randomly select an edge and delete it. This heuristic changes both the structure of the graph as well as the relation distribution across the KG. Note that the ATS metric will fail to recognize these perturbations to the KG by design, since after deletion the remaining edges are still valid triples .

Table 1: **Performance comparison on CSQA and OBQA perturbation experiments.** We report in-house test accuracy using the data split of Lin et al. (2019) on CSQA and the official test accuracy on OBQA. We perturb all the triples (100%) in the original KG.

	CSQA								OBQA							
	RN				MHGRN				RN				MHGRN			
Methods	Acc (↑)	ATS (↓)	SC2D (↓)	SD2 (↓)	Acc (↑)	ATS (↓)	SC2D (↓)	SD2 (↓)	Acc (↑)	ATS (↓)	SC2D (↓)	SD2 (↓)	Acc (↑)	ATS (↓)	SC2D (↓)	SD2 (↓)
w/o KG	53.41	-	-	-	53.41	-	-	-	62.00	-	-	-	62.00	-	-	-
w/ KG	56.87	0.940	-	-	57.21	0.940	-	-	66.80	0.934	-	-	68.00	0.934	-	-
Relation Swapping	53.42	0.831	0.168	6.20E-3	53.42	0.831	0.168	6.20E-3	67.00	0.857	0.189	7.79E-3	67.30	0.857	0.189	7.79E-3
Relation Replacement	53.42	0.329	**0.100**	1.70E-3	52.22	0.329	**0.100**	**1.70E-3**	66.80	0.269	**0.105**	1.84E-3	67.60	0.269	0.105	1.84E-3
Edge Rewiring	53.42	0.505	0.131	2.31E-3	52.22	0.505	0.131	2.31E-3	66.60	0.620	0.172	7.31E-3	67.00	0.620	0.172	7.31E-3
Edge Deletion	52.21	0.933	0.144	2.00E-3	51.00	0.933	0.144	2.00E-3	66.80	0.923	0.155	2.19E-3	67.60	0.923	0.155	2.19E-3
RL-RR	**55.21**	**0.322**	0.102	**1.66E-3**	**55.52**	**0.314**	0.101	1.78E-3	**67.30**	**0.255**	0.108	**1.79E-4**	**67.70**	**0.248**	**0.104**	**1.75E-4**

5.2 Evaluation Protocol and Metrics

As discussed above, the ATS metric is not sensitive to structure perturbation in the KG. Thus, we adopt the **Similarity on Clustering Coefficient Distribution (SC2D)** as a complementary metric. Given a KG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, we define $\mathcal{G}^r$ as the subgraph containing all the triples with a particular relation r, i.e., $\mathcal{G}^r = (\mathcal{E}, \mathcal{T}^{'})$ where $\mathcal{T}^{'} = \{(h, r', t) | h, t \in \mathcal{E}, r' = r\}$. We then compute the relation specific clustering coefficient vector $\mathbf{c^r}$ for each $\mathcal{G}^r$ where i^{th} element of $\mathbf{c^r}$ represents the local clustering coefficient (Watts & Strogatz, 1998) of the ith entity and compute the average clustering vector $\mathbf{c}$ for the whole graph as the average of $\mathbf{c^r}$ over all $r \in \mathcal{R}$, i.e., $\mathbf{c} = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} \mathbf{c^r}$. We denote the average clustering vector for the unperturbed KG and the perturbed KG by $\mathbf{c_o}$ and $\mathbf{c_p}$ respectively. We then define the SC2D metric as the reciprocal of the L2 distance between $\mathbf{c_o}$ and $\mathbf{c_p}$, i.e., $f^{SC2D}(\mathcal{G}, \mathcal{G}^{'}) = \frac{1}{\|\mathbf{c_o} - \mathbf{c_p}\|_2 + b}$, with b as a constant for avoiding zero division.

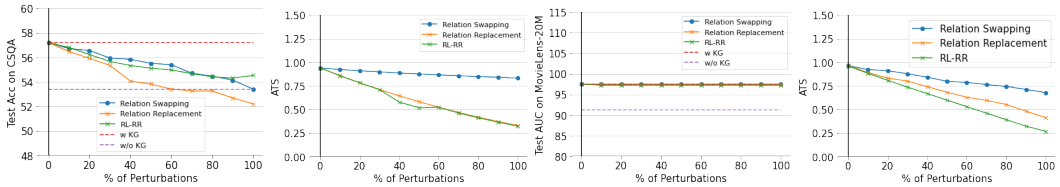
The SC2D metric is not effective for cases where the KG has a very small local clustering coefficient. To this end, we introduce another metric, **Similarity on Degree Distribution (SD2)**, which can also capture structural changes. Similar to the SC2D metric, we first create a set of graphs $\mathcal{G}^r$ for each relation $r \in \mathcal{R}$. We then calculate the relation specific degree vector $\mathbf{d^r}$ for each $\mathcal{G}^r$ where i^{th} element of $\mathbf{d^r}$ represents the degree of the ith node and calculate the average degree vector $\mathbf{d}$ for the whole graph as the mean of $\mathbf{d^r}$ over all $r \in \mathcal{R}$, i.e., $\mathbf{d} = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} \mathbf{d^r}$. We denote the average degree vector of the unperturbed KG and the perturbed KG by $\mathbf{d_o}$ and $\mathbf{d_p}$ respectively. We then define the SD2 metric of the KG as the reciprocal of the L2 distance between $\mathbf{d_o}$ and $\mathbf{d_p}$, i.e., $f^{SD2}(\mathcal{G}, \mathcal{G}^{'}) = \frac{1}{\|\mathbf{d_o} - \mathbf{d_p}\|_2 + b}$. In the later experiment we would validate these proposed evaluation metrics by comparing them with human evaluation.

5.3 Commonsense QA Experiments

We conduct experiments on CommonsenseQA (CSQA) and OpenBookQA (OBQA), both of which require external knowledge beyond context to answer the questions. The neural-symbolic QA systems we experiment with are MHGRN (§ 3) and Relation Networks (RN, Santoro et al. (2017)), which is a path-based model. MHGRN and RN are shown to significantly outperform baselines without KG in Feng et al. (2020). We train both models to replicate their original performance using BERT-Base as the text encoder, and freeze the models throughout the perturbation. The KG we aim to perturb is ConceptNet (Speer et al., 2016), a commonsense KG that fits to this task.

Results and Discussion. Table 1 shows our results of perturbing all the triples in KG. Note that perturbing relations or edges are not guaranteed to yield invalid/novel triples. Thus, a completely perturbed KG would still have a different degree of deviation from the original KG, depending on what perturbation method is used. This can be evidenced by the different evaluation metric scores in Table 1, which show that generally our RL-RR method always yield the perturbed KG with the most degree of deviation. Such a perturbed KG is not meaningless noise. This is supported by Table 3, which shows that our designed controlled noisy baselines (§ 3) lead to significant performance drop. Overall, the above observations demonstrate the necessity of introducing our RL-based perturbation method to complement the heuristic methods even when we do complete perturbation.

Moreover, we observe that using heuristic perturbation, both RN and MHGRN do not outperform the non-KG baseline on CSQA while on OBQA they manage to maintain their performance. Using RL-based targeted perturbation policy, both QA models achieve a statistically significant improvement according to the T-test over both heuristics and non-KG baselines, which demonstrates the superiority of our RL perturbation method over simple heuristics. Moreover, the success of perturbing KG completely while still preserving performance suggests both RN and MHGRN are relying



(a) Test Acc on CSQA (b) ATS on CSQA (c) AUC on MovieLens (d) SD2 on MovieLens

Figure 4: Variation of Performance and Evaluation Metrics with different scales of perturbation on MHGRN with CSQA and RippleNet with MovieLens-20M.

Metric	CSQA		OBQA	
	Val.	Int.	Val.	Int.
Ori. KG	0.4711	0.2089	0.5650	0.2050
RL-RR	0.4667	0.2756	0.3700	0.1950

Table 2: **Human evaluation** on the validity (Val.) and interpretability (Int.) of the paths from MHGRN.

Model	RN	MHGRN
w/ KG	56.87	57.21
w/ zero graph embedding	53.10	53.99
w/ random graph embedding	52.60	52.48

Table 3: Results of controlled noisy baselines on CSQA.

on problematic symbolic knowledge for prediction. Thus it validates our doubt on whether symbolic knowledge contributes to the improvement interpretably. Our later human study on the interpretability would further validates our doubt.

Figure 4 illustrates the change in performance of MHGRN on CSQA and the KG evaluation metrics across different scales of perturbation. We observe that both RL-RR and Relation Replacement heuristic are more effective in perturbing KG compared with Relation Swapping. We also observe that for larger scales of perturbations the RL policy can maintain the performance of the model better than the heuristics do, which validates the superiority of our learned RL policy.

Human Evaluation on Model Explanation. We conduct human evaluations on the symbolic paths given by MHGRN, which are claimed to serve as the explanation for MHGRN's prediction. We randomly select 10 questions from the CSQA and OBQA test sets which are predicted correctly by the model and obtain the top ranked paths that are attended most by MHGRN. We then ask 5 human annotators to annotate each path for its validity and interpretability. The average annotation scores is reported in Table 2 with a κ inter-annotator agreement as 0.29, which indicates a fair agreement. We observe that compared with the original KG, the paths from the two RL perturbed KGs contain more invalid symbolic knowledge, indicating our proposed RL perturbation is effective. Meanwhile, for all the KGs our annotators consider less than 28% of the paths can provide plausible explanation for the prediction. This validates our doubt on the faithfulness of explanation given by KG-based neural-symbolic models.

Quality of Evaluation Metrics. We also leverage the validity scores given by human to validate the effectiveness of our two proposed KG evaluation metrics, ATS and SC2D. The Pearson correlation coefficient between the human evaluation scores in Table 2 and the two KG evaluation metrics in Table 1 are 0.592 and 0.769 respectively, both showing high correlation. This demonstrates that the KG evaluation metrics well capture KG's semantic validity.

5.4 Recommender System Experiments

We conduct experiments on two recommender system benchmarks, Last.FM (Rendle, 2012) and MovieLens-20M dataset (Harper & Konstan, 2016). In particular, we use 20% of the train set for Last.FM and 40% of the train set for Movie Lens-20M to stimulate a cold start scenario, for which KGs are shown to be helpful (Wang et al., 2018b). We adopt KGCN and RippleNet (§ 3) as our neural-symbolic recommender system. Both models are trained on the datasets and then fixed throughout the perturbation. The item KGs we aim to perturb are constructed by Wang et al. (2019a) for Last.FM and MovieLens-20M respectively. Due to the sparse nature of these item KG, the local clustering coefficient of each node is extremely small which makes the SC2D metric meaningless. Therefore, we do not report the SC2D values in this section.

Results and analysis. In Table 4, we report the test AUC in % and the evaluation metrics after perturbing all the triples in the KG. We observe that despite a completely perturbed graph, the change in performance is statistically insignificant regardless of what type of perturbation we do. This

Table 4: **Performance comparison on recommender system datasets.** We report the test AUC for two KG-augmented models (KGCN and RippleNet) on Last.FM and MovieLens-20M. We perturb all the triples (100%) in the KG. Note that SD2C metric is not included as the local clustering coefficient is too small for the KG and thus does not provide useful insights.

Methods	**Last.FM**						**MovieLens 20M**					
	KGCN			**RippleNet**			**KGCN**			**RippleNet**		
	AUC (↑)	ATS (↓)	SD2 (↓)	AUC (↑)	ATS (↓)	SD2 (↓)	AUC (↑)	ATS (↓)	SD2 (↓)	AUC (↑)	ATS (↓)	SD2 (↓)
w/o KG	50.75	-	-	50.75	-	-	91.30	-	-	91.30	-	-
w/ KG	55.99	0.972	-	56.23	0.972	-	96.62	0.960	-	97.46	0.960	-
Relation Swapping	55.98	0.681	3.05E-2	56.23	0.681	3.05E-2	**96.62**	0.678	6.14E-4	**97.46**	0.678	6.14E-4
Relation Replacement	55.98	0.415	0.339	56.22	0.415	0.339	96.50	0.413	**7.74E-5**	97.45	0.413	**7.74E-5**
Edge Rewiring	55.98	0.437	1.89E-2	56.22	0.437	1.89E-2	96.24	0.679	4.44E-4	96.62	0.679	4.44E-4
Edge Deletion	55.96	0.941	3.38E-3	56.21	0.941	3.38E-3	96.36	0.982	1.02E-4	97.22	0.982	1.02E-4
RL-RR	**56.04**	**0.320**	**1.3E-3**	**56.28**	**0.310**	**1.2E-3**	96.53	**0.401**	2.23E-4	97.25	**0.268**	2.21E-4

strongly suggests that it is the (false) connection between entities instead of the semantic stored in the KG that leads to the improvement over non-KG baseline. Thus, the neural-symbolic recommender system built on top of these false connection could hardly provide any faithful explanation.

Figure 4 shows how the performance and the KG evaluation metric change across differnt scale of perturbation with KGCN on Last.FM dataset. We observe insignificant change in performance (flat curve) regardless of the perturbation methods while the drastic drop in ATS validates the semantic of KG is indeed largely perturbed. This again strongly suggests that the models are not relying on any faithful symbolic knowledge to conduct prediction.

6 Related Work

Knowledge Graph Augmented Models. In many recently proposed tasks, it has been shown that performance of the system can be improved with the introduction of external knowledge. Currently Knowledge Graphs (KGs) have become the most common source of external knowledge that systems try to leverage to improve their performance. Even the successes of pretrained Language models in the field of NLP have been hypothesized as the ability of such LM's to probe a latent knowledge base (Petroni et al., 2019). In the past KGs have been used in various tasks such as Commonsense Question Answering (Lin et al., 2019; Shen et al., 2020; Lv et al., 2020; Musa et al., 2019), Recommendations (Wang et al., 2019b; 2020; Song et al., 2019; Cao et al., 2019), Natural Language Inference (Wang et al., 2019c) and many others (Chen et al., 2019; Kapanipathi et al.), greatly boosting model's performance compared to non-KG methods. KGs have also been regarded as an important way to make the model more interpretable and generate explanations (Lin et al., 2019; Zhang et al., 2019; Song et al., 2019; Cao et al., 2019; Gao et al., 2019; Ai et al., 2018).

Perturbation and Attacks on Graphs. Inspired by Computer Vision and NLP, there have recently been quite a few works that investigate into the problem of Adversarial Perturbations in the Graph Learning framework (Chen et al., 2020). Though these works try to perturb the graph, the fundamental problem that they solve is very different from the one we try to solve. These works try to adversarially perturb the graph minimally to have maximum impact on the performance whereas our goal is to use perturbations to analyze KG augmented models. These works use a variety of techniques to perturb the KG ranging from gradient based(Chen et al., 2018; Bojchevski & Günnemann, 2019; Wu et al., 2019), RL based(Ma et al., 2019; Dai et al., 2018), AutoEncoder based(Chen et al., 2018) and more. Moreover very few existing works on graph perturbations have worked on heterogeneous graphs such as Knowledge Graphs. (Ma et al., 2019) use a Reinforcement Learning based adversarial perturbation policy, but they only consider homogeneous graphs in their framework whereas we have a KG which is heterogeneous in nature. To the best of our knowledge, this is the first work that considers perturbations for analyzing KG augmented models.

7 Conclusion

In this paper, we analyze the structures learned by a KG augmented neuro-symbolic model via targeted KG perturbations. We show that simple heuristics are enough to show that the structures learnt by the model are insensitive to changes in the KG which can lead to corrupt explanations being generated by such models. We also introduce a RL based perturbation framework that tries to deceive the model into learning structures such that even with a corrupt KG we get a significant improvement over the non-KG baseline. Human evaluation on the explanations generated by the models with our perturbed KG shows a clear degradation in the quality of the generated explanations.

REFERENCES

Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms*, 11(9):137, 2018.

Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *International Conference on Machine Learning*, pp. 695–704. PMLR, 2019.

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*, pp. 151–161, 2019.

Daoyuan Chen, Yaliang Li, Min Yang, Hai-Tao Zheng, and Ying Shen. Knowledge-aware textual entailment with graph attention network. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019.

Jinyin Chen, Yangyang Wu, Xuanheng Xu, Yixian Chen, Haibin Zheng, and Qi Xuan. Fast gradient attack on network embedding. *arXiv preprint arXiv:1809.02797*, 2018.

Liang Chen, Jintang Li, Jiaying Peng, Tao Xie, Zengxu Cao, Kun Xu, Xiangnan He, and Zibin Zheng. A survey of adversarial learning on graphs. *arXiv preprint arXiv:2003.05730*, 2020.

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.

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.

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, pp. 3622–3629, 2019.

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.

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

Pavan Kapanipathi, Veronika Thost, Siva Sankalp Patel, Spencer Whitehead, Ibrahim Abdelaziz, Avinash Balakrishnan, Maria Chang, Kshitij P Fadnis, R Chulaka Gunasekara, Bassem Makni, et al. Infusing knowledge into the textual entailment task using graph convolutional networks.

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

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)*, pp. 2822–2832, 2019.

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.

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

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.

Ryan Musa, Xiaoyan Wang, Achille Fokoue, Nicholas Mattei, Maria Chang, Pavan Kapanipathi, Bassem Makni, Kartik Talamadupula, and Michael Witbrock. Answering science exam questions using query reformulation with background knowledge. In *Automated Knowledge Base Construction (AKBC)*, 2019. URL https://openreview.net/forum?id=HJxYZ-5paX.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

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

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*, pp. 4967–4976, 2017.

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*, pp. 593–607. Springer, 2018.

Tao Shen, Yi Mao, Pengcheng He, Guodong Long, Adam Trischler, and Weizhu Chen. Exploiting structured knowledge in text via graph-guided representation learning. *ArXiv*, abs/2004.14224, 2020.

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.

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

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

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*, pp. 592–600, 2018a.

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*, pp. 417–426, 2018b.

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*, pp. 968–977, 2019a.

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

Xiang Wang, Yao-Kun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. Reinforced negative sampling over knowledge graph for recommendation. *Proceedings of The Web Conference 2020*, 2020.

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, pp. 7208–7215, 2019c.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998. ISSN 0028-0836. doi: 10.1038/30918. URL http://dx.doi.org/10.1038/30918.

Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples on graph data: Deep insights into attack and defense. *arXiv preprint arXiv:1903.01610*, 2019.

Wen Zhang, Bibek Paudel, Wei Zhang, Abraham Bernstein, and Huajun Chen. Interaction embeddings for prediction and explanation in knowledge graphs. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019.