SalKG: Learning From Knowledge Graph Explanations for Commonsense Reasoning

Aaron Chan[♣], **Soumya Sanyal**[♣*], **Boyuan Long**[♣*],
Jiashu Xu[♣*], **Tanishq Gupta**[◇*†], **Xiang Ren**[♣]
[♣]University of Southern California, [◇]IIT Delhi
{chanaaro, soumyasa, boyuanlo, jiashuxu, xiangren}@usc.edu,
Tanishq.Gupta.mt617@maths.iitd.ac.in

Abstract

Augmenting pre-trained language models with knowledge graphs (KGs) has achieved success on various commonsense reasoning tasks. Although some works have attempted to explain the behavior of such KG-augmented models by indicating which KG inputs are salient (i.e., important for the model's prediction), it is not always clear how these explanations should be used to make the model better. In this paper, we explore whether KG explanations can be used as supervision for teaching these KG-augmented models how to filter out unhelpful KG information. To this end, we propose SalKG, a simple framework for learning from KG explanations of both coarse (*Is the KG salient?*) and fine (*Which parts of the KG are salient?*) granularity. Given the explanations generated from a task's training set, SalKG trains KG-augmented models to solve the task by focusing on KG information highlighted by the explanations as salient. Across two popular commonsense QA benchmarks and three KG-augmented models, we find that SalKG's training process can consistently improve model performance. [1]

1 Introduction

To function well in the real world, natural language processing (NLP) systems generally need common sense (Gunning, 2018). However, reporting bias for commonsense knowledge makes it hard for pre-trained language models (PLMs) (Devlin et al., 2019; Liu et al., 2019) to learn common sense from corpora alone (Davis and Marcus, 2015; Marcus, 2018). Although PLMs may capture some commonsense knowledge (Davison et al., 2019; Petroni et al., 2019), it is usually insufficient for tasks that require substantial commonsense reasoning (CSR) (Feng et al., 2020; Wang et al., 2019).

In contrast to corpora, knowledge graphs (KGs) are a rich and structured source of commonsense knowledge, containing numerous facts of the form `(concept1, relation, concept2)`. As a result, many methods have adopted the *KG-augmented model* paradigm, which combines a PLM with a graph encoder that reasons over the KG (Fig. 1). By incorporating information from the KG, such models have outperformed PLMs on a number of CSR tasks, including question answering (QA) (Lin et al., 2019; Feng et al., 2020).

Some KG-augmented models are also designed to explain their predictions by indicating which KG inputs are salient (i.e., important for the model's prediction). However, it is not necessarily obvious how these KG explanations should be used, especially for improving the model. Recently, Pruthi et al. (2020) proposed using text saliency explanations to regularize a PLM's self-attention mechanism, then evaluating explanations by their ability to improve the PLM's performance. Inspired by this idea, we view KG explanations as rich signals for teaching KG-augmented models how to filter out task-irrelevant KG information.

Unlike text, KGs' symbolic properties naturally support explanations at different levels of KG granularity (e.g., graph, path, node). Thus, we propose *Learning from **Sal**iency Explanations of **KG**-Augmented Models* (**SalKG**), a simple framework for learning from both coarse (graph-level) and fine (node/path-level) explanations of KG saliency. Given explanations created from a task's training set, SalKG trains KG-augmented models to solve the task by attending to KG inputs highlighted by the explanations as salient. This encourages the model to ignore KG information that is not helpful for solving the task. Across two popular commonsense QA benchmarks, three KG-augmented models, and two PLMs, SalKG can yield consistent performance gains over both non-SalKG-trained KG-augmented models and PLMs.

*Equal contribution.
†Work done while TG interned remotely at USC.
[1]Code will be released at github.com/INK-USC/SalKG.

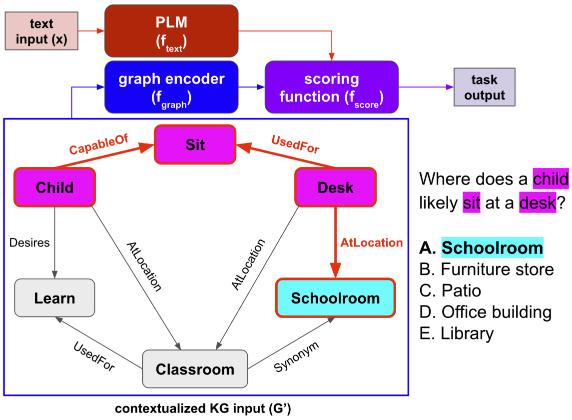

Figure 1: **KG-Augmented Commonsense QA.** KG-augmented models use both text and KG inputs to solve CSR tasks like commonsense QA. Given a question and candidate answer choice, a contextualized KG ($\mathcal{G}'$) is constructed from the full KG by using concepts in the question-answer text. However, $\mathcal{G}'$ may sometimes contain irrelevant information, which should be filtered out by the model.

2 Background

CSR is the ability to perceive, understand, or judge things, using basic knowledge that is shared by almost all humans but rarely stated explicitly (Gunning, 2018). In this work, we consider NLP tasks that are designed to require CSR (e.g., QA) (Talmor et al., 2019; Mihaylov et al., 2018). To solve CSR tasks, we use KG-augmented models, in which a PLM is augmented with a graph encoder — e.g., graph neural network (GNN) — for KG reasoning (Fig. 1).

Basic Notation Given some CSR task, we define x as the text input for the task and $\mathcal{F}$ as the model. A KG is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$, where $\mathcal{V}$, $\mathcal{R}$, and $\mathcal{E}$ are the KG's nodes (concepts), relations, and edges (facts), respectively. An edge is a directed triple of the form $e = (h, r, t) \in \mathcal{E}$, in which $h \in \mathcal{V}$ is the head node, $t \in \mathcal{V}$ is the tail node, and $r \in \mathcal{R}$ is the relation between h and t. A path is a connected sequence of edges in the KG.

When solving a CSR task instance, the model does not use the entire KG, since most information in $\mathcal{G}$ will be irrelevant to x. Instead, the model uses a smaller, contextualized KG $\mathcal{G}' = (\mathcal{V}', \mathcal{R}', \mathcal{E}')$, which is built from $\mathcal{G}$ with respect to x. Usually, this is done heuristically by extracting $\mathcal{V}' \subseteq \mathcal{V}$ as the concepts mentioned in x, $\mathcal{R}' \subseteq \mathcal{R}$ as the relations between concepts in $\mathcal{V}'$, and $\mathcal{E}' \subseteq \mathcal{E}$ as the edges involving $\mathcal{V}'$ and $\mathcal{R}'$ (Lin et al., 2019; Feng et al., 2020). If $\mathcal{G}$ does not provide enough relevant information to extract a good $\mathcal{G}'$, then new edges are sometimes added to $\mathcal{G}'$ using a PLM-based generator (Bosselut et al., 2019; Wang et al., 2020; Yan et al., 2020). Since the model never uses the full KG at once, in the rest of the paper, we will use "KG" to refer to the contextualized KG $\mathcal{G}'$.

KG-Augmented Models As introduced in previous work (Lin et al., 2019; Feng et al., 2020), a KG-augmented model $\mathcal{F}_{\text{KG}}$ has three main components: PLM f_{text}, graph encoder f_{graph}, and scoring function f_{score}. First, $\mathbf{x} = f_{\text{text}}(x)$ is the embedding of text input x, where f_{text} is a PLM text encoder (Devlin et al., 2019; Liu et al., 2019). Second, $\mathbf{g} = f_{\text{graph}}(\mathcal{G}', \mathbf{x})$ is the embedding of contextualized KG $\mathcal{G}'$. KG-augmented models mainly differ in their design of $\mathcal{F}_{\text{graph}}$, reasoning over $\mathcal{G}'$ via message passing (Schlichtkrull et al., 2018; Feng et al., 2020) or edge/path aggregation (Lin et al., 2019; Bosselut and Choi, 2019; Ma et al., 2019). Third, $\mathcal{F}_{\text{KG}}(x, \mathcal{G}') = f_{\text{score}}([\mathbf{x}, \mathbf{g}])$ is the KG-augmented model's task output, where f_{score} is usually a multi-layer perceptron (MLP), and $[\cdot, \cdot]$ denotes concatenation. Also, let $\mathcal{F}_{\text{No-KG}}(x) = f_{\text{score}}(\mathbf{x})$ denote the task output for a PLM that does not use $\mathcal{G}'$.

2.1 Commonsense QA

While KG-augmented models are applicable to any KG-related CSR task (e.g., natural language inference), we consider multi-choice QA, which is one of the most popular CSR tasks (Sap et al., 2020; Storks et al., 2019). Given a question q, set of answer choices $A = \{a_i\}$, and target answer $a^* \in A$, the QA model's objective is to predict a confidence probability $p(q, a_i)$ for each (q, a_i) pair, so that $a^* = \arg\max_{a_i \in A} p(q, a_i)$.

To use KG-augmented models for multi-choice QA, we set $x = (q, a_i)$, build $\mathcal{G}'_i$ using concepts mentioned in (q, a_i), and compute (q, a_i)'s probability as $p_{\text{KG}}(q, a_i) = \mathcal{F}_{\text{KG}}(x, \mathcal{G}'_i)$. To use PLMs for multi-choice QA, we instead compute (q, a_i)'s probability as $p_{\text{No-KG}}(q, a_i) = \mathcal{F}_{\text{No-KG}}(x)$. Also, let Φ be some evaluation metric for our CSR task.

3 KG Saliency Explanations

In this section, we describe how KG saliency explanations are created. Given a model and task instance, we consider an *explanation* to be a subset of inputs that is identified as being most salient to the model's prediction for this instance. Such explanations are sometimes called extractive explanations (Wiegreffe et al., 2020). Although KG-augmented

Explanation Setting	Saliency Unit
SALKG-Coarse	KG
SALKG-Fine (MHGRN)	Node
SALKG-Fine (RN)	Path
SALKG-Fine (PathGen)	Path

Table 1: Summary of saliency unit types used for different explanation settings, each characterized by saliency mode (e.g., coarse, fine) and graph encoder (e.g., MHGRN, RN, PathGen). Sec. 5 has more information about these three graph encoders.

models take both text and KG inputs, in this work, we limit our analysis to explanations of KG inputs.

We define two modes for measuring the saliency of KG inputs: *coarse saliency* scores the importance of the KG (i.e., $\mathcal{G}'$) as a whole, while *fine saliency* scores the importance of individual components within the KG. Also, we define a *saliency unit* as a KG input being scored. Using coarse or fine saliency explanations created from the training set, SALKG (Sec. 4) trains KG-augmented models to focus on the task-relevant saliency units.

3.1 Coarse Saliency Explanations

Coarse saliency units are graphs — specifically, KGs. Given a task instance, coarse saliency involves deciding between using the KG (i.e., use KG-augmented model's prediction) or not using the KG (i.e., use PLM's prediction), which depends on how much relevant information the KG provides for this instance.

The coarse saliency score for instance x, with respect to some KG $\mathcal{G}'$, is measured as the performance difference between the KG-augmented model's prediction and the PLM's prediction. That is, given KG-augmented model output $\mathcal{F}_{\text{KG}}(x, \mathcal{G}')$, PLM output $\mathcal{F}_{\text{No-KG}}(x)$, and evaluation metric Φ, the general form of coarse saliency score is:

$$s_{\text{c}}(x) = \Phi(\mathcal{F}_{\text{KG}}(x, \mathcal{G}')) - \Phi(\mathcal{F}_{\text{No-KG}}(x))$$

The notion of coarse saliency described above is task-agnostic. However, to instantiate coarse saliency with respect to the multi-choice QA task, we define a specific form in which the coarse saliency score is computed for each answer choice. As described in Sec. 2, a multi-choice QA instance consists of question q, answer choices $A = \{a_i\}$, and target answer $a^* \in A$. Also, for $\mathcal{F}_{\text{KG}}$, recall that a KG $\mathcal{G}'_i$ is built for each (q, a_i) pair, so that (q, a_i)'s probability is computed as $p_{\text{KG}}(q, a_i) = \mathcal{F}_{\text{KG}}(x, \mathcal{G}'_i)$. In light of this, we want a model that predicts higher probabilities for $a_i = a^*$ and lower probabilities for $a_i \neq a^*$. Intuitively, $\mathcal{G}'_i$ is more salient if using $\mathcal{F}_{\text{KG}}$ instead of $\mathcal{F}_{\text{No-KG}}$ increases the predicted probability for $a_i = a^*$ and decreases the predicted probability for $a_i \neq a^*$. Thus, the coarse saliency score of answer choice a_i is computed as:

$$s_{\text{c}}(q, a_i) = \begin{cases} p_{\text{KG}}(q, a_i) - p_{\text{No-KG}}(q, a_i), & a_i = a^* \\ p_{\text{No-KG}}(q, a_i) - p_{\text{KG}}(q, a_i), & a_i \neq a^* \end{cases}$$

After computing $s_{\text{c}}(q, a_i)$ for each (q, a_i), we create coarse saliency explanations by binarizing (q, a_i)'s KG $\mathcal{G}'_i$ as salient (positive) if $s_{\text{c}}(q, a_i) > T$ or not salient (negative) if $s_{\text{c}}(q, a_i) \leq T$, given some threshold T. Let $y_{\text{c}}(x) = y_{\text{c}}(q, a_i)$ denote the binarized version of $s_{\text{c}}(x) = s_{\text{c}}(q, a_i)$. More details are in Appendix A.1.

3.2 Fine Saliency Explanations

Fine saliency units are either nodes or paths. Since each graph encoder reasons over graphs by updating/pooling a certain type of graph component, it is natural to consider the saliency of this graph component type when analyzing the given graph encoder. Thus, we define a graph encoder's saliency units as the KG components being updated and pooled into its graph embedding $\mathbf{g}$. For example, in the MHGRN graph encoder (Feng et al., 2020), graph embedding $\mathbf{g}$ is obtained by updating/pooling the node embeddings, so MHGRN's saliency units are nodes. We refer to graph encoders using node and path units as node-based and path-based graph encoders, respectively.

Fine saliency assumes at least part of the KG is used and involves indicating whether each saliency unit should be used for predicting a given instance. This can be done by computing a fine saliency score for every saliency unit in the KG. While fine saliency can be computed using any off-the-shelf feature attribution method, as a proof of concept, we use the well-known *gradient*×*input* (G×I) method (Li et al., 2015; Denil et al., 2014). Given saliency unit u and its embedding $\mathbf{u} \in \mathbb{R}^d$ in $\mathcal{G}'$, the general form of the G×I fine saliency score is:

$$s_{\text{f}}(u) = \sum_{j=1}^{d} \mathbf{u}_j \frac{\partial F_{\text{KG}}(x, \mathcal{G}')}{\partial \mathbf{u}_j}$$

For multi-choice QA, u is a more salient unit in $\mathcal{G}'_i$ if it increases $p_{\text{KG}}(q, a_i)$ for $a_i = a^*$ and decreases $p_{\text{KG}}(q, a_i)$ for $a_i \neq a^*$. Thus, we compute

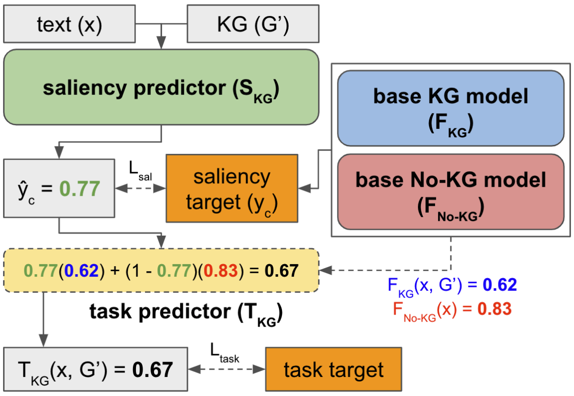

Figure 2: **SALKG-Coarse Model.** This model solves the CSR task by choosing between the two base models' ($\mathcal{F}_{\text{KG}}$ and $\mathcal{F}_{\text{No-KG}}$) predictions for each instance. During training, the model is supervised by coarse saliency targets (y_c), which explain whether the KG is salient for different instances.

$s_f(\mathbf{u})$ in multi-choice QA as follows:

$$s_f(u) = \begin{cases} \sum_{j=1}^{d} \mathbf{u}_j \frac{\partial p_{\text{KG}}(q,a_i)}{\partial \mathbf{u}_j}, & a_i = a^* \\ -\sum_{j=1}^{d} \mathbf{u}_j \frac{\partial p_{\text{KG}}(q,a_i)}{\partial \mathbf{u}_j}, & a_i \neq a^* \end{cases}$$

After computing $s_f(\mathbf{u})$ for each $\mathbf{u}$ in $\mathcal{G}'_i$, we create fine saliency explanations by binarizing the top-k-scoring saliency units in $\mathcal{G}'_i$ as salient (positive) and the rest as not salient (negative). Let $y_f(\mathbf{u})$ denote the binarized version of $s_f(\mathbf{u})$. More details are in Appendix A.2.

4 SALKG

By default, KG-augmented models always use the KG (i.e., $\mathcal{G}'$). However, the KG sometimes provides little to no relevant information for the given task instance. Hence, the model's performance may suffer if the model is not explicitly taught when the KG or certain parts of the KG are likely to be salient. To address this limitation, the SALKG framework enables models to learn about KG saliency from explanations created using procedures described in Sec. 3. Specifically, SALKG trains KG-augmented models to solve the task by attending to KG inputs highlighted by the explanations as salient, which encourages the model to ignore KG information that is not helpful for solving the task. In this section, we first describe the base model upon which SALKG is built, then discuss how SALKG trains a stronger, "saliency-aware" model using coarse and fine saliency explanations.

4.1 Base Models

The base models are KG-augmented model $\mathcal{F}_{\text{KG}}$ and PLM $\mathcal{F}_{\text{No-KG}}$, which are used for some CSR task of interest (e.g., multi-choice QA). First, both base models are trained on the chosen task. Second, the trained $\mathcal{F}_{\text{KG}}$ (and $\mathcal{F}_{\text{No-KG}}$, if doing coarse saliency) is used to compute a real-valued score for each saliency unit. This score indicates how salient the unit is to the model's prediction on a given instance. We consider both coarse (s_c) and fine (s_f) saliency scores, which indicate the saliency of KGs and nodes/paths, respectively. Third, we create an explanation for each saliency unit by discretizing its saliency score into a binary label. We also call these labels as *saliency targets*, since they serve as learning targets for the SALKG model.

4.2 SALKG Model

At a high level, the SALKG model $\mathcal{F}^*_{\text{KG}}$ consists of two components: a saliency predictor $\mathcal{S}_{\text{KG}}$ and a task predictor $\mathcal{T}_{\text{KG}}$. Because creating explanations requires ground truth labels for the CSR task, explanations are not available to the model at test time. Thus, using saliency explanations created from the training set, $\mathcal{S}_{\text{KG}}$ is trained to predict which KG inputs are likely to be salient for a given task instance. Then, $\mathcal{T}_{\text{KG}}$ uses $\mathcal{S}_{\text{KG}}$'s saliency predictions to output a prediction for the CSR task. Below, we describe how $\mathcal{F}^*_{\text{KG}}$ is implemented for coarse (SALKG-Coarse) and fine (SALKG-Fine) saliency.

SALKG-Coarse In Sec. 3.1, we discussed how KG information may be helpful for some instances, but distracting for others. We also showed how binary coarse saliency targets y_c are created to explain when the KG is helpful to the model. Given instance x, $y_c = 1$ if KG is salient, and $y_c = 0$ if KG is not salient. Ideally, our model would use the KG-augmented model if the KG is salient, but use the PLM if the KG is not salient.

SALKG-Coarse aims to do this by forming an adaptive ensemble of KG-augmented base model $\mathcal{F}_{\text{KG}}$ and PLM $\mathcal{F}_{\text{No-KG}}$, with each model's weight predicted by $\mathcal{S}_{\text{KG}}$ with respect to x. That is, $\mathcal{S}_{\text{KG}}$ is a binary MLP classifier trained to predict y_c from x. Let $\hat{y}_c = \mathcal{S}_{\text{KG}}(x, \mathcal{G}') \in [0, 1]$ be $\mathcal{S}_{\text{KG}}$'s saliency prediction for x. Then, we define $\mathcal{T}_{\text{KG}}$'s task prediction as:

$$\mathcal{T}_{\text{KG}}(x, \mathcal{G}') = \hat{y}_c\, \mathcal{F}_{\text{KG}}(x, \mathcal{G}') + (1 - \hat{y}_c)\, \mathcal{F}_{\text{No-KG}}(x),$$

where $\mathcal{T}_{\text{KG}}(x, \mathcal{G}')$ is the final output of SALKG-Coarse. $\mathcal{F}^*_{\text{KG}}$'s learning objective is $\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda \mathcal{L}_{\text{sal}}$, where $\mathcal{L}_{\text{task}}$ is the CSR task loss, $\mathcal{L}_{\text{sal}}$ is cross entropy loss for saliency prediction, and $\lambda \geq 0$ is a loss weighting parameter. Note that $\mathcal{F}_{\text{KG}}$ and

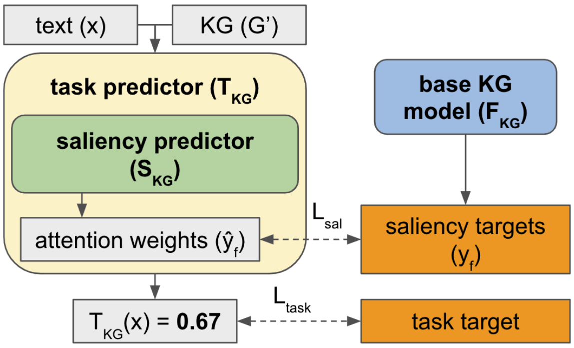

Figure 3: **SALKG-Fine Model.** This model is trained to solve the CSR task by attending to relevant nodes/paths in the KG. During training, the model is supervised by fine saliency targets (y_f), which explain which nodes/paths are salient for different instances.

$\mathcal{F}_{\text{No-KG}}$ are not jointly trained with $\mathcal{S}_{\text{KG}}$ and $\mathcal{T}_{\text{KG}}$. After initially training $\mathcal{F}_{\text{KG}}$ and $\mathcal{F}_{\text{No-KG}}$ on the CSR task, we save their predictions and explanations to disk, then discard the two models.

SALKG-Fine In Sec. 3.2, we discussed how each KG saliency unit (e.g., node, path) can either be helpful or distracting for a given instance. We also showed how a binary fine saliency target y_{f} is created to explain if a saliency unit is helpful to the model. Given unit u, $y_f = 1$ if u is salient, and $y_f = 0$ if u is not salient. Ideally, our model would use focus more on units for which $y_f = 1$.

In state-of-the-art graph encoders, all saliency unit embeddings within a KG are pooled into a single graph embedding via multi-head self-attention. Thus, we train the SALKG-Fine model to upweight salient units by regularizing its graph encoder's attention weights to approximate the fine saliency targets. For SALKG-Fine, $\mathcal{F}^*_{\text{KG}}$ has the same architecture as $\mathcal{F}_{\text{KG}}$. Here, $\mathcal{S}_{\text{KG}}$ is $\mathcal{F}_{\text{KG}}$'s self-attention mechanism, which predicts each unit's saliency target, and $\mathcal{T}_{\text{KG}}$ is $\mathcal{F}_{\text{KG}}$'s task predictor MLP, which directly predicts the CSR task labels as $\mathcal{T}_{\text{KG}}(x, \mathcal{G}')$. Thus, $\mathcal{F}^*_{\text{KG}}$ is again trained using the following objective: $\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda\mathcal{L}_{\text{sal}}$, where $\mathcal{L}_{\text{sal}}$ is still cross entropy loss.

5 Experiments

In this section, we evaluate the performance of SALKG-trained models on commonsense reasoning tasks. Through our experiments, we show that SALKG-trained models can consistently outperform non-SALKG-trained KG-augmented models and PLMs.

Datasets We test SALKG on the CommonsenseQA (CSQA) and OpenbookQA (OBQA) datasets, which are two of the most popular benchmarks for commonsense multi-choice QA. We report performance using test accuracy, which is the standard evaluation metric for multi-choice QA. For CSQA, we follow the commonly used protocol of reporting test accuracy on the in-house data split from Lin et al. (2019), as official test labels are not publicly available. Furthermore, following prior work in commonsense multi-choice QA, we use ConceptNet (Speer et al., 2017) as our KG $\mathcal{G}$.

KG-Augmented Models We experiment with KG-augmented models composed from all combinations of two PLMs and three graph encoders. For PLMs, we consider RoBERTa(-Large) (Liu et al., 2019) and BERT(-Base) (Devlin et al., 2019). For graph encoders, we use MHGRN (Feng et al., 2020), PathGenerator (PathGen) (Wang et al., 2020), and Relation Network (RN) (Santoro et al., 2017; Lin et al., 2019). MHGRN is a node-based graph encoder, while PathGen and RN are path-based graph encoders.

Saliency Target Thresholds In Sec. 3, we stated that the discretization of real-valued saliency scores into binary saliency targets depends on the values of certain threshold parameters. In all of our experiments, we use the same thresholds. For SALKG-Coarse on multi-choice QA, the threshold T depends on the base PLM's ($\mathcal{F}_{\text{No-KG}}$) predicted answer choice probabilities for the given question instance. See Appendix A.1 for more details about how T is computed. For SALKG-Fine, we use $k = 10$ for top-k selection. This threshold was chosen by manually inspecting the saliency of 50 randomly sampled answer choice KGs in CSQA.

5.1 Main Results

In Tables 2 and 3, we demonstrate that models trained using the SALKG framework can yield improved performance on CSQA and OBQA. First, rows 1-2 report the performance of our two baselines: PLMs (No-KG) and non-SALKG-trained KG-augmented base models (KG). Second, rows 3-4 show the performance of our proposed SALKG-Coarse and SALKG-Fine, respectively. Third, row 5-6 display the performance of SALKG-Coarse-Target and SALKG-Fine-Target, respectively. These are variants of SALKG-Coarse and SALKG-Fine where the saliency targets are directly used, rather than predicted. Since using the

Model	CSQA Test Accuracy (%)					
	MHGRN		PathGen		RN	
	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa
No-KG	55.44	70.59	55.44	70.59	55.44	70.59
KG	56.68	73.33	56.65	72.04	55.60	71.07
SALKG-Coarse	**57.37** (±0.08)	**74.03** (±0.09)	**58.23** (±0.82)	**72.93** (±0.29)	**58.15** (±0.05)	**73.73** (±0.45)
SALKG-Fine	55.44 (±1.81)	73.46 (±0.69)	53.34 (±1.75)	70.99 (±0.85)	56.05 (±0.54)	69.59 (±1.26)
SALKG-Coarse-Target	66.16	82.60	68.57	80.10	67.45	79.69
SALKG-Fine-Target	66.85 (±5.33)	77.14 (±3.14)	81.57 (±0.36)	82.06 (±4.12)	79.10 (±0.74)	79.85 (±10.47)

Table 2: **CSQA Performance Comparison.** SALKG-Coarse outperforms both baselines (No-KG and KG) on all model settings, whereas SALKG-Fine only beats $\max(\text{No-KG}, \text{KG})$ on RoBERTa+MHGRN and BERT+RN. Both SALKG-Coarse and SALKG-Fine still lag far behind their respective oracle models, SALKG-Coarse-Target and SALKG-Fine-Target.

saliency targets requires the test labels, SALKG-Coarse-Target and SALKG-Fine-Target are oracle models and cannot be directly compared to the models in rows 1-4. SALKG-Coarse-Target and SALKG-Fine-Target are only meant to provide an upper bound for the performance of SALKG-Coarse and SALKG-Fine. Note that only models within the same column should be compared. More details about the oracle models and the evaluation protocol can be found in Appendices A.3 and A.4, respectively.

If SALKG models are learning effectively from saliency explanations, then they should outperform $\max(\text{No-KG}, \text{KG})$, the maximum accuracy across both baselines. Below, we discuss specific results on CSQA and OBQA.

CSQA Table 2 shows our results on CSQA. First, we see that KG outperforms No-KG on all settings, which supports the finding in prior works that KG-augmented models generally improve performance on CSQA. The average improvement from No-KG to KG is 1.21%. Second, we find that SALKG-Coarse beats $\max(\text{No-KG}, \text{KG})$ on all settings by a slightly larger average margin of 1.51%. Evidently, SALKG-Coarse models are learning effectively from coarse saliency explanations. Third, we observe that SALKG-Fine is not quite as effective, only beating the baselines on RoBERTa+MHGRN and BERT+RN. For BERT+PathGen, SALKG-Fine actually performs much worse than $\min(\text{No-KG}, \text{KG})$, achieving a performance drop of 2.10%.

Furthermore, the SALKG-Coarse-Target oracle outperforms SALKG-Coarse by an average of 8.36%. Meanwhile, the SALKG-Fine-Target oracle beats SALKG-Fine by an even larger average of 14.62%. This indicates much opportunity for both SALKG-Coarse and SALKG-Fine to be trained better. Also, SALKG-Fine-Target's training seems to be unstable, with standard deviation accuracy of up to 10.47 on RoBERTa+RN.

OBQA Table 3 shows our results on OBQA. First, we see that KG outperforms No-KG on all but two settings, which again supports the finding in prior works that KG-augmented models generally improve performance on OBQA. The average improvement from No-KG to KG is 1.43%. Second, we find that SALKG-Coarse beats $\max(\text{No-KG}, \text{KG})$ on all but one setting (BERT+RN) by a slightly smaller average margin of 1.09%. In particular, for BERT+MHGRN and BERT+PathGen, SALKG-Coarse beats $\max(\text{No-KG}, \text{KG})$ by impressive margins of 2.27% and 3.07%, respectively. Like in CSQA, it is clear that SALKG-Coarse models are learning effectively from coarse saliency explanations. Third, we observe that SALKG-Fine only beats $\max(\text{No-KG}, \text{KG})$ on BERT+MHGRN and is surprisingly outperformed by $\min(\text{No-KG}, \text{KG})$ by as much as 8.20% (RoBERTa+MHGRN).

Furthermore, the SALKG-Coarse-Target oracle outperforms SALKG-Coarse by an average of 9.58%. Meanwhile, SALKG-Fine-Target outperforms SALKG-Fine by an average of 8.03%. Strangely, on RoBERTa+RN, SALKG-Fine-Target's accuracy is only 52.07%, which is 15.13% lower than SALKG-Fine's. Somehow, directly using the saliency targets for RoBERTa+RN hurts performance dramatically. Currently, the exact reason for this is unclear to us and requires further investigation. Overall, this again shows that there is still much room for SALKG-Coarse and SALKG-Fine to improve.

Analysis For both CSQA and OBQA, SALKG-Coarse consistently outperforms $\max(\text{No-KG}, \text{KG})$, showing that learning to

Model	OBQA Test Accuracy (%)					
	MHGRN		PathGen		RN	
	BERT	RoBERTa	BERT	RoBERTa	BERT	RoBERTa
No-KG	53.60	68.40	53.60	68.40	53.60	68.40
KG	53.20	69.80	55.00	67.80	**58.60**	70.20
SALKG-Coarse	**55.87** (±1.10)	**69.93** (±0.42)	**58.07** (±0.92)	**69.20** (±0.40)	57.93 (±0.50)	**71.13** (±0.50)
SALKG-Fine	55.07 (±1.27)	60.20 (±7.00)	51.53 (±1.36)	67.47 (±1.14)	51.20 (±1.40)	67.20 (±1.71)
SALKG-Coarse-Target	70.60	79.40	65.00	76.60	69.00	79.00
SALKG-Fine-Target	72.27 (±7.60)	78.40 (±7.79)	60.33 (±0.58)	71.27 (±0.50)	66.53 (±3.40)	52.07 (±3.56)

Table 3: **OBQA Performance Comparison.** SALKG-Coarse outperforms both baselines (No-KG and KG) on all model settings except BERT+RN, while SALKG-Fine only beats $\max$(No-KG, KG) on BERT+MHGRN. Overall, both SALKG-Coarse and SALKG-Fine are still outperformed by their respective oracle models, SALKG-Coarse-Target and SALKG-Fine-Target. However, SALKG-Fine-Target's performance is unexpectedly low for RoBERTa+RN and requires further investigation.

Model	CSQA Test Accuracy (%)
No-KG	55.44
KG	56.65
SALKG-Coarse	58.23 (±0.82)
SALKG-Coarse (Pipeline)	56.14 (±0.44)
SALKG-Coarse (No Pos. Weight)	55.44 (±0.00)
SALKG-Coarse (Self-Attention)	58.37 (±0.09)
SALKG-Fine	53.34 (±1.75)
SALKG-Fine (KL Divergence)	53.48 (±1.93)

Table 4: **Ablation Studies.** Using BERT+PathGen on CSQA, we perform ablations of SALKG's design choices along various dimensions: training procedure, loss weighting, coarse saliency model architecture, and fine saliency loss function.

predict coarse saliency explanations is an effective training strategy. In Sec. 5.3 and Table 5, we further analyze why SALKG-Coarse is able to work well.

On the other hand, SALKG-Fine's results are not quite as strong. These negative results could be caused by multiple reasons. One possible reason is the design of $\mathcal{F}^*_{\text{KG}}$, which includes $\mathcal{S}_{\text{KG}}$'s architecture, $\mathcal{T}_{\text{KG}}$'s architecture, and $\mathcal{F}^*_{\text{KG}}$'s loss function. Another reason could be the creation of the fine saliency explanations, which includes the choices of feature attribution method, saliency target binarization method, and threshold parameter k. We plan to improve SALKG-Fine by exploring these two directions in future work.

5.2 Ablation Studies

In Table 4, we present several ablation studies to validate the effectiveness of our design choices in SALKG-Coarse and SALKG-Fine. In all ablation studies, we use BERT+PathGen on CSQA. In the top section, we again show the baseline models' performance. In the middle section, we show the performance of SALKG-Coarse and its ablation variants. In the bottom section, we show the performance of SALKG-Fine and its ablation variants. Each ablation variant is denoted with its ablation name in parentheses.

Joint Training of $\mathcal{S}_{\text{KG}}$ and $\mathcal{T}_{\text{KG}}$ For SALKG-Coarse, we train $\mathcal{S}_{\text{KG}}$ and $\mathcal{T}_{\text{KG}}$ jointly by default. For SALKG-Coarse, $\mathcal{T}_{\text{KG}}$ is just a weighted sum function, so joint training means $\mathcal{S}_{\text{KG}}$ is trained using the CSR task loss with respect to $\mathcal{T}_{\text{KG}}$'s output. Alternatively, we could train $\mathcal{S}_{\text{KG}}$ and $\mathcal{T}_{\text{KG}}$ sequentially in a pipeline, such that $\mathcal{S}_{\text{KG}}$ and $\mathcal{T}_{\text{KG}}$ do not share any learning signal. In Table 4, we see that SALKG-Coarse greatly outperforms SALKG-Coarse (Pipeline) by 2.09%, which suggests that $\mathcal{S}_{\text{KG}}$ is best trained using both the CSR task loss and saliency prediction loss.

Upweighting Positive Saliency Instances For both CSQA and OBQA, we observed high class imbalance in the binarized coarse saliency targets. On average, each dataset has approximately 30 times more negative targets than positive targets. By default, when training SALKG-Coarse, we scale the losses of positive saliency instances by a fixed weight equal to the ratio of negative instances to positive instances (i.e., around 30). For SALKG-Coarse (No Pos. Weight), all instances' losses are weighted equally, which significantly hurts the performance by 2.79%. This shows that our positive loss weighting strategy is beneficial.

Coarse Saliency Predictor In SALKG-Coarse, $\mathcal{S}_{\text{KG}}$ is implemented as an MLP by default. Besides MLP, we also considered implementing $\mathcal{S}_{\text{KG}}$ as a multi-head self-attention mechanism, following SALKG-Fine's implementation of $\mathcal{S}_{\text{KG}}$. We find that SALKG-Coarse (Self-Attention) performs only 0.14% better than SALKG-Coarse. While 0.14% is actually a bit higher than SALKG-Coarse (Self-Attention)'s standard deviation of 0.09%, it is still much lower than SALKG-Coarse's standard deviation of 0.82%, which means we can conclude that SALKG-Coarse and SALKG-Coarse (Self-Attention) perform about equally.

Fine Saliency Loss In SALKG-Fine, we implement $\mathcal{S}_{\text{KG}}$ as a binary classifier and train $\mathcal{S}_{\text{KG}}$ using binary cross entropy (BCE) loss. By doing so, we treat each saliency unit as an individual saliency instance. Besides, BCE loss, we also tried training $\mathcal{S}_{\text{KG}}$ with KL divergence. Here, the model aims to minimize the KL divergence between the $\mathcal{S}_{\text{KG}}$'s predicted attention weight distribution and the saliency target distribution, which means all saliency units in the KG together form a single saliency instance. We see that SALKG-Fine (KL Divergence) only beats SALKG-Fine by 0.14%, which is much lower than both models' reported standard deviations. Thus, we conclude that SALKG-Fine (KL Divergence) and SALKG-Fine have comparable performance.

5.3 Coarse Saliency Analysis

SALKG-Coarse is based on the insight that KG information may help the model on some instances but hurt on others. Thus, even if KG outperforms No-KG on average, No-KG may still correctly predict some instances that KG got wrong. SALKG-Coarse takes advantage of such complementary predictions between No-KG and KG, in order to achieve performance higher than $\max(\text{No-KG}, \text{KG})$. As shown by BERT+MHGRN and RoBERTa+PathGen on OBQA (Table 3), SALKG-Coarse can still beat $\max(\text{No-KG}, \text{KG})$ even when No-KG outperforms KG.

In Table 5, we analyze the performance of BERT and/or PathGen on various sets of questions in CSQA. Through this performance breakdown, we can isolate the potential improvement contributed by each base model to SALKG-Coarse. We begin by looking at the questions for which SALKG-Coarse has no influence. These are the 46.01% of questions correctly answered by both models and the 33.92% of questions incorrectly answered by both models. Since SALKG-Coarse is trained to choose between the two models' predictions, SALKG-Coarse's output is fixed if both models make the same prediction. This leaves 20.07% of questions that were correctly answered by exactly one of the two models: 9.43% were from No-KG, while the other 10.43% were from KG. This 20.07% of constitutes the complementary predictions leveraged by SALKG-Coarse.

Question Set	Question Percentage (%)
No-KG Correct	55.44
KG Correct	56.65
Only No-KG Correct	9.43
Only KG Correct	10.64
Both Correct	46.01
Both Incorrect	33.92
At Least One Incorrect	66.08
SALKG-Coarse Correct	58.23 (±0.82)
SALKG-Coarse-Target Correct	68.57

Table 5: **Coarse Saliency Analysis.** Using BERT+PathGen on CSQA, we present a performance breakdown for various question sets, in order to analyze why SALKG-Coarse is able to outperform No-KG and KG.

Based on this question-level analysis, we estimate the SALKG-Coarse-Target oracle accuracy to be 66.08%, the percentage of questions that at least one model answered correctly. However, as stated in Sec. 3.1, coarse saliency targets are created at the answer choice level (not question level), which offers us more flexibility to choose between No-KG and KG. As a result, SALKG-Coarse-Target's accuracy is actually 68.57%. This leaves SALKG-Coarse (58.23%) significant room for improvement, perhaps through better model architecture and training.

6 Related Work

6.1 Creating Model Explanations

Text-Based Explanations Many works have been proposed for explaining the predictions of language models, especially PLMs. Although some of these works focus on abstractive (free-text) explanations (Rajani et al., 2019; Strout et al., 2019; Zhao and Vydiswaran, 2020), most aim to provide extractive explanations which highlight salient tokens in the model's text input. Such extractive explanations typically use either gradient-based (Sundararajan et al., 2017; Li et al., 2015; Denil et al., 2014), attention-based (Mohankumar et al., 2020; Tutek and Šnajder, 2020; Ghaeini et al., 2018; Lee et al., 2017), and occlusion-based (DeYoung et al., 2019; Poerner et al., 2018; Kádár et al., 2017; Li et al., 2016) feature attribution methods. How feature attribution methods should be chosen remains an open question and the subject of much recent debate (Bastings and Filippova, 2020; Wiegreffe and Pinter, 2019; Serrano and Smith, 2019; Jain and Wallace, 2019). While SALKG also uses feature attribution methods (e.g., G×I) to create extractive explanations, our study is limited to explanations regarding KG-augmented models' graph inputs.

Graph-Based Explanations There are also methods proposing extractive explanations for graph encoders, especially GNNs. Such explanations are designed to point out components in the graph input that contribute most to the model's prediction. Some GNNs use attention for pooling, which naturally highlights nodes with higher attention weights (Lee et al., 2019, 2018). More sophisticated approaches use post-hoc optimization to identify salient nodes (Huang et al., 2020; Ying et al., 2019) or subgraphs (Ying et al., 2019).

Unlike individual PLMs and graph encoders, KG-augmented models take both text and graph inputs. The KG-augmented model's graph encoder usually computes graph embeddings via attention pooling of nodes/paths, and the attention weights can be used to explain which nodes/paths in the input KG are salient (Lin et al., 2019; Feng et al., 2020; Liu et al., 2020; Wang et al., 2020; Yan et al., 2020). These KG explanations can be interpreted as identifying knowledge in the KG that is complementary to the knowledge encoded in the PLM. However, there is little work on how such KG explanations should be used. SALKG considers graph-based extractive explanations of KG-augmented models, but focuses more on how explanations are used rather than created.

6.2 Learning From Model Explanations

To improve the model's learning, explanations can be used in a diverse range of ways, including as extra supervision or regularization (Pruthi et al., 2020; Hase et al., 2020; Narang et al., 2020; Andreas et al., 2017), pruned inputs (Jain et al., 2020; Bastings et al., 2019; Lei et al., 2016), additional inputs (Hase and Bansal, 2021; Co-Reyes et al., 2018), and intermediate variables (Wiegreffe et al., 2020; Zhou et al., 2020; Rajani et al., 2019). The most similar work to ours is Pruthi et al. (2020), which proposed using extractive text explanations to regularize a PLM's self-attention mechanism and demonstrated considerable performance gains. Still, methods for learning from explanations have largely focused on domains like text and images, as opposed to graphs. To the best of our knowledge, SALKG is the first work to train KG-augmented models using KG explanations as supervision.

7 Conclusion

In this paper, we investigated the usage of KG explanations as supervision for teaching KG-augmented models how to filter out task-irrelevant KG information. To this end, we proposed SALKG, a simple framework for learning from KG explanations of both coarse and fine granularity. Across two popular commonsense QA benchmarks and three KG-augmented models, we found that SALKG-Coarse almost always outperformed our baselines, while SALKG-Fine showed less consistent improvement. In future work, we plan to explore several directions for improving SALKG-Fine's design as well as ideas for combining SALKG-Coarse and SALKG-Fine into a hybrid framework.

References

Jacob Andreas, Dan Klein, and Sergey Levine. 2017. Learning with latent language. *arXiv preprint arXiv:1711.00482*.

Jasmijn Bastings and Katja Filippova. 2020. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? *arXiv preprint arXiv:2010.05607*.

Joost Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. *arXiv preprint arXiv:1905.08160*.

Antoine Bosselut and Yejin Choi. 2019. Dynamic knowledge graph construction for zero-shot commonsense question answering. *arXiv preprint arXiv:1911.03876*.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

John D Co-Reyes, Abhishek Gupta, Suvansh Sanjeev, Nick Altieri, Jacob Andreas, John DeNero, Pieter Abbeel, and Sergey Levine. 2018. Guiding policies with language via meta-learning. *arXiv preprint arXiv:1811.07882*.

Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103.

Joe Davison, Joshua Feldman, and Alexander M Rush. 2019. Commonsense knowledge mining from pretrained models. 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 1173–1178.

Misha Denil, Alban Demiraj, and Nando De Freitas. 2014. Extraction of salient sentences from labelled documents. *arXiv preprint arXiv:1412.6815*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C Wallace. 2019. Eraser: A benchmark to evaluate rationalized nlp models. *arXiv preprint arXiv:1911.03429*.

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

Reza Ghaeini, Xiaoli Z Fern, and Prasad Tadepalli. 2018. Interpreting recurrent and attention-based neural models: a case study on natural language inference. *arXiv preprint arXiv:1808.03894*.

David Gunning. 2018. Machine common sense concept paper. *arXiv preprint arXiv:1810.07528*.

Peter Hase and Mohit Bansal. 2021. When can models learn from explanations? a formal framework for understanding the roles of explanation data. *arXiv preprint arXiv:2102.02201*.

Peter Hase, Shiyue Zhang, Harry Xie, and Mohit Bansal. 2020. Leakage-adjusted simulatability: Can models generate non-trivial explanations of their behavior in natural language? *arXiv preprint arXiv:2010.04119*.

Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. 2020. Graphlime: Local interpretable model explanations for graph neural networks. *arXiv preprint arXiv:2001.06216*.

Sarthak Jain and Byron C Wallace. 2019. Attention is not explanation. *arXiv preprint arXiv:1902.10186*.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C Wallace. 2020. Learning to faithfully rationalize by construction. *arXiv preprint arXiv:2005.00115*.

Akos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43(4):761–780.

Jaesong Lee, Joong-Hwi Shin, and Jun-Seok Kim. 2017. Interactive visualization and manipulation of attention-based neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 121–126.

John Boaz Lee, Ryan Rossi, and Xiangnan Kong. 2018. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1666–1674.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-attention graph pooling. In *International Conference on Machine Learning*, pages 3734–3743. PMLR.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of EMNLP-IJCNLP*, pages 2829–2839, Hong Kong, China. Association for Computational Linguistics.

Ye Liu, Tao Yang, Zeyu You, Wei Fan, and Philip S Yu. 2020. Commonsense evidence generation and injection in reading comprehension. *arXiv preprint arXiv:2005.05240*.

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

Kaixin Ma, Jonathan Francis, Quanyang Lu, Eric Nyberg, and Alessandro Oltramari. 2019. Towards generalizable neuro-symbolic systems for commonsense question answering. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pages 22–32, Hong Kong, China. Association for Computational Linguistics.

Gary Marcus. 2018. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

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

Akash Kumar Mohankumar, Preksha Nema, Sharan Narasimhan, Mitesh M Khapra, Balaji Vasan Srinivasan, and Balaraman Ravindran. 2020. Towards transparent and explainable attention models. *arXiv preprint arXiv:2004.14243*.

Sharan Narang, Colin Raffel, Katherine Lee, Adam Roberts, Noah Fiedel, and Karishma Malkan. 2020. Wt5?! training text-to-text models to explain their predictions. *arXiv preprint arXiv:2004.14546*.

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

Nina Poerner, Benjamin Roth, and Hinrich Schütze. 2018. Evaluating neural network explanation methods using hybrid documents and morphological agreement. *arXiv preprint arXiv:1801.06422*.

Danish Pruthi, Bhuwan Dhingra, Livio Baldini Soares, Michael Collins, Zachary C Lipton, Graham Neubig, and William W Cohen. 2020. Evaluating explanations: How much do explanations from the teacher aid students? *arXiv preprint arXiv:2012.00893*.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. *arXiv preprint arXiv:1906.02361*.

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

Maarten Sap, Vered Shwartz, Antoine Bosselut, Yejin Choi, and Dan Roth. 2020. Commonsense reasoning for natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 27–33.

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

Sofia Serrano and Noah A Smith. 2019. Is attention interpretable? *arXiv preprint arXiv:1906.03731*.

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

Shane Storks, Qiaozi Gao, and Joyce Y Chai. 2019. Recent advances in natural language inference: A survey of benchmarks, resources, and approaches. *arXiv preprint arXiv:1904.01172*.

Julia Strout, Ye Zhang, and Raymond J Mooney. 2019. Do human rationales improve machine explanations? *arXiv preprint arXiv:1905.13714*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Martin Tutek and Jan Šnajder. 2020. Staying true to your word:(how) can attention become explanation? *arXiv preprint arXiv:2005.09379*.

Peifeng Wang, Nanyun Peng, Pedro Szekely, and Xiang Ren. 2020. Connecting the dots: A knowledgeable path generator for commonsense question answering. *arXiv preprint arXiv:2005.00691*.

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

Sarah Wiegreffe, Ana Marasovic, and Noah A Smith. 2020. Measuring association between labels and free-text rationales. *arXiv preprint arXiv:2010.12762*.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. *arXiv preprint arXiv:1908.04626*.

Jun Yan, Mrigank Raman, Aaron Chan, Tianyu Zhang, Ryan Rossi, Handong Zhao, Sungchul Kim, Nedim Lipka, and Xiang Ren. 2020. Learning contextualized knowledge structures for commonsense reasoning. *arXiv preprint arXiv:2010.12873*.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. In *Advances in neural information processing systems*, pages 9244–9255.

Xinyan Zhao and VG Vydiswaran. 2020. Lirex: Augmenting language inference with relevant explanation. *arXiv preprint arXiv:2012.09157*.

Wangchunshu Zhou, Jinyi Hu, Hanlin Zhang, Xiaodan Liang, Maosong Sun, Chenyan Xiong, and Jian Tang. 2020. Towards interpretable natural language understanding with explanations as latent variables. *arXiv preprint arXiv:2011.05268*.

A Appendix

A.1 Coarse Saliency Explanations

Across different questions, the base PLM's ($\mathcal{F}_{\text{No-KG}}$) predicted answer choice distribution may vary significantly. For some questions, there may be multiple answers with very similar probabilities. For other questions, one answer's probability may dominate the rest. Thus, it may not make sense to use the same fixed threshold for all questions in the dataset.

For multi-choice QA, we designed the threshold T to depend on the base PLM's ($\mathcal{F}_{\text{No-KG}}$) answer choice probabilities for the given question instance. For some question instance, let $p^{(1)}$ and $p^{(2)}$ be the highest and second-highest probabilities among $\mathcal{F}_{\text{No-KG}}$'s predicted answer choice probabilities. Also, let T' be a scaling parameter. Then, the threshold T for the given instance is computed as: $T = T'(p^{(1)} - p^{(2)})$. T' was chosen such that SALKG-Coarse-Target's validation accuracy is maximized.

A.2 Fine Saliency Explanations

Depending on the graph encoder, the saliency unit may or may not be given to the model as a single embedding. While node-based graph encoders take node embeddings as input, path-based graph encoders do not take path embeddings as input. Instead path-based graph encoders take node and relation embeddings as input, then form path embeddings from these node and relation embeddings. As a result, the computation of fine saliency scores is slightly different between node-based and path-based graph encoders.

For node-based encoders, the saliency unit embeddings $\mathbf{u}$ are just the node embeddings. Thus, a node's fine saliency score is computed directly using the $s_f(u)$ equation given in Sec. 3.2.

For path-based encoders, given a path, we first use the $s_f(u)$ equation to compute a separate fine saliency score for each node embedding and relation embedding in the path. Then, we compute the path saliency score as the sum of the scores of its constituent nodes and relations.

A.3 Oracle SALKG Models

Here, we describe the two oracle SALKG models in more detail. Given the coarse saliency targets, SALKG-Coarse-Target simply involves choosing the "correct" prediction (between the two base models) for each saliency instance. This process runs very quickly and does not require additional training. On the other hand, SALKG-Fine-Target involves training the SALKG-Fine model while applying the fine saliency targets as a binary mask to the attention weights, prior to softmax.

A.4 Evaluation Protocol

For fair comparison between models, we adapt the following evaluation protocol. First, for No-KG, we choose the model configuration with the highest mean validation accuracy, over three seeds. However, among these three seeds, only one seed's model checkpoint can be used as a base No-KG model for SALKG. Thus, as the base No-KG model, we select the seed whose model's validation accuracy is highest and only report this single-seed test accuracy in row 1. For KG, the same procedure is done to obtain the base KG model and the result in row 2. Second, the SALKG models are built upon the single-seed base models obtained in the first step. For each SALKG model, we report the mean and standard deviation of the test accuracy over three seeds. Although only single-seed test accuracy is reported for the baselines, we actually maintain a fair comparison between the baselines and SALKG, since the same base models are used in both cases. For this reason, it would not be fair to compare the baselines' three-seed average accuracy to the SALKG models' three-seed average accuracy.

For SALKG-Coarse-Target, we only report a single accuracy value, since it is fully deterministic. For SALKG-Fine-Target, we report mean and standard deviation of the test accuracy over three seeds, since it involves additional model training.