Learning Contextualized Knowledge Structures for Commonsense Reasoning

Jun Yan[1], Mrigank Raman[2], Aaron Chan[1], Tianyu Zhang[3], Ryan Rossi[4], Handong Zhao[4], Sungchul Kim[4], Nedim Lipka[4], Xiang Ren[1]
University of Southern California[1], Indian Institute of Technology Delhi[2], Tsinghua University[3], Adobe Research[4]
{yanjun, chanaaro, xiangren}@usc.edu, {mt1170736}@iitd.ac.in,
{zhang-ty17}@mails.tsinghua.edu.cn,
{ryrossi, hazhao, sukim, lipka}@adobe.com

Abstract

Recently, neural-symbolic models have achieved noteworthy success in leveraging knowledge graphs (KGs) for commonsense reasoning tasks, like question answering (QA). However, fact sparsity, inherent in human-annotated KGs, can hinder such models from retrieving task-relevant knowledge. To address these issues, we propose **Hybrid Graph Network (HGN)**, a neural-symbolic model that reasons over both extracted (human-labeled) and generated facts within the same *learned* graph structure. Given a KG subgraph of extracted facts, HGN is jointly trained to generate complementary facts, encode relational information in the resulting "hybrid" subgraph, and filter out task-irrelevant facts. We demonstrate HGN's ability to produce contextually pertinent subgraphs by showing considerable performance gains across four commonsense reasoning benchmarks and a user study of fact validness and helpfulness. [1]

1 Introduction

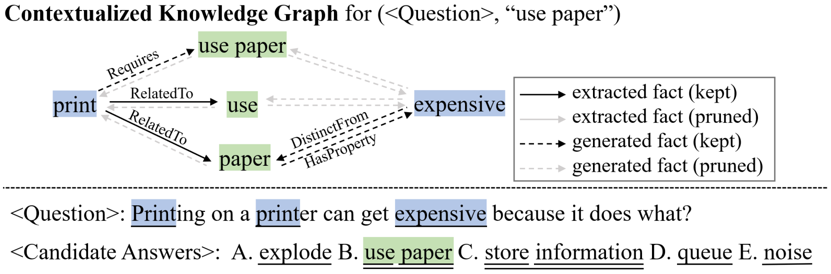
Common sense is essential for natural language understanding (NLU) systems to function effectively in the real world (Apperly, 2010). Yet, since commonsense knowledge is self-evident to humans, it is rarely stated in natural language (Gunning, 2018). This makes it difficult for neural pre-trained language models (PLMs) (Devlin et al., 2019; Liu et al., 2019) to learn common sense from corpora alone (Davis and Marcus, 2015; Marcus, 2018). Fig. 1 illustrates how such unstated knowledge may be needed in NLU tasks. To answer the question in Fig. 1, one must incorporate auxiliary knowledge about relations between pertinent concepts — *e.g.*, the fact that *printing requires using paper*. While humans naturally possess this knowledge, PLMs usually have to obtain it from external sources (Lin et al., 2019; Feng et al., 2020; Malaviya et al., 2020; Bosselut and Choi, 2019).

Many recent commonsense reasoning methods (Lin et al., 2019; Feng et al., 2020; Malaviya et al., 2020; Bosselut and Choi, 2019) follow the paradigm of augmenting PLMs with external symbolic knowledge from a KG (*e.g.*, ConceptNet (Speer et al., 2017)), which consists of organized concepts (nodes), relations, and facts (edges). In contrast to corpora, KGs can provide structured commonsense facts of the form (*concept1*, *relation*, *concept2*). This allows models to make interpretable predictions via complex multi-hop reasoning over the KG (Lin et al., 2019; Feng et al., 2020; Wang et al., 2020).

Despite the growing success of this paradigm, obtaining relevant KG facts for a given task instance remains a great challenge. Existing methods (Lin et al., 2019; Feng et al., 2020) generally assume that the KG's predefined facts are sufficient for the given task. However, KGs are constructed using human annotations, so they inherently omit numerous relevant facts and may even contain noisy facts. To address this fact sparsity issue, Wang et al. (2020) propose using a PLM to generate a set of relational paths (*i.e.*, sequences of connected facts) from a KG-extracted subgraph. However, this method may itself generate irrelevant facts and offers no mechanism for pruning them. Furthermore, Wang et al. (2020) largely ignore graph structure (*i.e.*, connectivity) and does not facilitate interaction between extracted and generated facts.

In response to these limitations, we propose **Hybrid Graph Network (HGN)**, a neural-symbolic model that reasons over both extracted and generated facts within the same *learned* graph structure. Given a subgraph of extracted facts, HGN is jointly trained to generate complementary facts, encode inter-fact relations in the resulting "hybrid" sub-

[1] The code for our experiments has been uploaded and will be published.



<Question>: Printing on a printer can get expensive because it does what?

<Candidate Answers>: A. explode B. use paper C. store information D. queue E. noise

Figure 1: **KG-Augmented Commonsense QA.** Predicting the correct answer ("use paper") requires commonsense facts like (print, Requires, use paper) and (paper, HasProperty, expensive), which are not given in the question and candidate answers. HGN uses facts extracted from the KG, *e.g.*, (print, RelatedTo, use), but also generates new facts, eventually keeping relevant ones, *e.g.*, (print, Requires, use paper) and (paper, HasProperty, expensive), while pruning irrelevant ones, *e.g.*, (use, · , expensive).

graph, and filter out task-irrelevant facts (Fig. 1). Unlike prior methods, HGN learns to contextualize the subgraph's structure for the task instance, so that knowledge propagation between facts is adaptively controlled. HGN achieves this by leveraging two novel architectural components: **(1)** a *learned adjacency matrix*, in which each edge has a weight; and **(2)** *edge-weighted message passing*, in which an edge's weight is used to regulate information flow along the edge.

In this paper, we show that HGN is able to generate KG subgraphs that include helpful facts and exclude unhelpful ones, leading to improved performance on three commonsense QA benchmarks. Plus, we present a user study demonstrating that humans find HGN facts to be more valid and helpful than facts yielded by a strong baseline method.

2 Problem Formulation

Our paper focuses on methods that augment PLMs with a symbolic KG, and we refer to such methods as neural-symbolic KG (NSKG) models. Here, given some NLU task, let x be the text input and y be the model's output. We denote a KG as $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$. $\mathcal{V}$, $\mathcal{R}$, and $\mathcal{E}$ are the sets of concepts (*a.k.a.* entities, nodes, vertices), relations, and facts (*a.k.a.* edges), respectively, in the KG. A fact is a directed triple of the form $e = (h, r, t) \in \mathcal{E}$, where $h \in \mathcal{V}$ is the head concept, $t \in \mathcal{V}$ is the tail concept, and $r \in \mathcal{R}$ is the relation between h and t. Also, let $\oplus$ indicate the concatenation operation.

A NSKG model has three main components: a text encoder f_{text}, a graph encoder f_{graph}, and a multilayer perceptron (MLP) classifier f_{MLP}. First, $\mathbf{s} = f_{\text{text}}(x; \theta_{\text{text}})$ denotes the encoding of x, where f_{text} tends to be a Transformer PLM (Devlin et al., 2019; Liu et al., 2019). Second, as supporting evidence, we extract a x-specific subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{R}', \mathcal{E}')$ from the KG $\mathcal{G}$ (Fig. 1). Typically, this is done heuristically by selecting $\mathcal{V}' \subseteq \mathcal{V}$ 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 facts involving $\mathcal{V}'$ and $\mathcal{R}'$. $\mathbf{g} = f_{\text{graph}}(\mathcal{G}', \mathbf{s}; \theta_{\text{graph}})$ then denotes the joint encoding of $\mathcal{G}'$ and $\mathbf{s}$. Third, the output is computed as $y = f_{\text{MLP}}(\mathbf{s} \oplus \mathbf{g}; \theta_{\text{MLP}})$. Finally, let $\theta = \{\theta_{\text{text}}, \theta_{\text{graph}}, \theta_{\text{MLP}}\}$ be the NSKG model's learnable parameters. NSKG models primarily differ in their design of f_{graph}, and HGN embodies the proposal of several key improvements to f_{graph}.

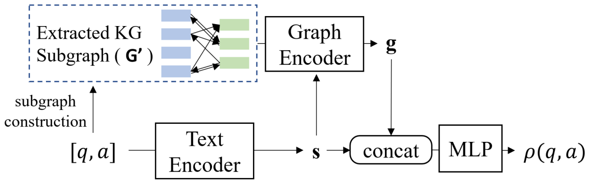


Figure 2: **Architecture of a typical neural-symbolic model for commonsense reasoning.**

In this work, we consider the multiple-choice commonsense QA task, although HGN can be easily adapted to other reasoning tasks (*e.g.*, natural language inference). Given a question q and set of candidate answers $\{a_i\}$, the QA model's goal is to predict a plausibility score $\rho(q, a)$ for each $a \in \{a_i\}$, so that the highest score is predicted for the correct answer. To use HGN for commonsense QA, we set $x = (q, a)$ and $y = \rho(q, a)$. The NSKG QA pipeline is illustrated in Fig. 2.

3 Hybrid Graph Network (HGN)

Because of KG fact sparsity, it can be difficult for NSKG models to obtain task-relevant facts from the KG. Like other NSKG models, HGN heuristically extracts an instance-specific subgraph from the KG, as described in §2. However, other models generally assume the extracted subgraph's facts are sufficient and ignore the fact sparsity issue (Lin et al., 2019; Feng et al., 2020). While Wang et al. (2020) addresses fact sparsity by generating new fact paths and reasoning over them, these generated facts can be irrelevant yet cannot be pruned. Moreover, Wang et al. (2020) largely disregards graph structure and inhibits interaction between extracted and generated facts.

Conversely, HGN reasons over a hybrid subgraph of both extracted and generated facts, while softly pruning any irrelevant facts. Unlike other NSKG models, HGN learns to modify the extracted subgraph's structure, iteratively re-weighting its interconnected facts in a way that best solves the given task instance. This is realized via HGN's *learned adjacency matrix*, where each edge has a weight, and *edge-weighted message passing*, where an edge's weight is used to regulate information flow along the edge. To encourage better edge pruning, we further impose entropy regularization on the learned edge weights.

3.1 KG Subgraph Notation

Building on §2, consider a (q, a)-specific subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{R}', \mathcal{E}')$ extracted from $\mathcal{G}$. $\mathcal{G}'$ has n_q question nodes (concepts), n_a answer nodes, and $n = n_q + n_a$ total nodes. Let $\mathcal{V}'_q$ and $\mathcal{V}'_a$ be the question and answer nodes, respectively, such that $\mathcal{V}' = \mathcal{V}'_q \cup \mathcal{V}'_a = \{v_1, \dots, v_n\}$. $\mathcal{G}'$ also has m edges (facts), which are denoted as $\mathcal{E}' = \{e_1, \dots, e_m\}$.

Let $V = \{\mathbf{v}_i \in \mathbb{R}^{d_v} \mid v_i \in \mathcal{V}'\}$ be $\mathcal{G}'$'s node embeddings, $R = \{\mathbf{r}_i \in \mathbb{R}^{d_r} \mid r_i \in \mathcal{R}'\}$ be its relation embeddings, $\mathbf{V} \in \mathbb{R}^{n \times d_v}$ be its node embedding matrix, and $\mathbf{E} \in \mathbb{R}^{n \times n \times d_e}$ be its edge embedding tensor. $\mathbf{E}_{(i,j)} \in \mathbb{R}^{d_e}$ is the edge embedding for node pair (v_i, v_j) if there exists an edge embedding for (v_i, v_j), and $\mathbf{E}_{(i,j)} = \mathbf{0}^{d_e}$ otherwise. Note that $\mathbf{E}$ initially consists only of edge embeddings for the m extracted facts provided in $\mathcal{G}'$.

Define $\mathbf{A} \in \{0, 1\}^{n \times n}$ as the node adjacency matrix of $\mathcal{G}'$. We only consider edges for question-answer node pairs, so let $\mathcal{V}'_{qa} = (\mathcal{V}'_q \times \mathcal{V}'_a) \cup (\mathcal{V}'_a \times \mathcal{V}'_q)$ be the set of question-answer node pairs in $\mathcal{V}'$. Then, $\mathbf{A}_{(i,j)} = 1$ if $(v_i, v_j) \in \mathcal{V}'_{qa}$, and $\mathbf{A}_{(i,j)} = 0$ otherwise. Observe that $\mathbf{A}$ currently contains m ones, and $m \leq 2n_q n_a$.

3.2 Model Overview

HGN begins by using a PLM to generate an edge embedding for each of the $\bar{m} = 2n_q n_a - m$ unlabeled node pairs, thus populating $\mathbf{E}$ with $2n_q n_a$ total edge embeddings. Now, every one-valued $\mathbf{A}_{(i,j)}$ has a corresponding embedding vector in $\mathbf{E}$. As a result, we have a hybrid subgraph of m extracted facts and $\bar{m}$ generated facts.

HGN reasons over the hybrid subgraph by iteratively updating $\mathbf{V}$, $\mathbf{E}$, and $\mathbf{A}$. HGN's L-layer reasoning module replaces all ones in $\mathbf{A}$ with continuous edge attention weights $\mathbf{A}_{(i,j)} \in [0, 1]$, which are updated together with $\mathbf{V}$ and $\mathbf{E}$ at each layer $\ell \in \{1, 2, ..., L\}$. Crucially, the edge weights in $\mathbf{A}$ are used to regulate edge-to-node message passing for updating $\mathbf{V}$. Meanwhile, node-to-edge message passing is used to update $\mathbf{E}$ and $\mathbf{A}$. We denote $(\mathbf{V}, \mathbf{E}, \mathbf{A})$ at layer ℓ as $(\mathbf{V}^\ell, \mathbf{E}^\ell, \mathbf{A}^\ell)$. After reasoning is done, HGN computes the graph embedding $\mathbf{g}$ by pooling the node embeddings in $\mathbf{V}^L$ and edge embeddings in $\mathbf{E}^L$.

In the following subsections, we describe each component of HGN in greater detail. First, we explain how the hybrid subgraph is constructed (§3.3). Next, we explain how HGN reasons over the hybrid subgraph by updating $\mathbf{V}$, $\mathbf{E}$, and $\mathbf{A}$ (§3.4). Last, we explain HGN's learning objective, particularly the entropy regularization of $\mathbf{A}^L$ (§3.5).

3.3 Hybrid Subgraph Construction

3.3.1 Node Embeddings

The first step of retrieving knowledge from $\mathcal{G}$ is concept grounding, which involves identifying text spans in (q, a) that match nodes in $\mathcal{V}$. Each node $v \in \mathcal{V}'$ is represented by an embedding $\mathbf{v} \in V$, which can be initialized using off-the-shelf methods like BERT (Devlin et al., 2019) and TransE (Bordes et al., 2013).

3.3.2 Edge Embeddings

After concept grounding, we need to obtain edge embeddings for all node pairs in $\mathcal{V}'_{qa}$. $\mathcal{G}'$ provides extracted facts about m of the $2n_q n_a$ question-answer node pairs. Any extracted fact $(v_i, r, v_j) \in \mathcal{E}'$ has a labeled relation $r \in \mathcal{R}'$ and off-the-shelf relation embedding $\mathbf{r} \in R$ (Bordes et al., 2013), so we initialize (v_i, r, v_j)'s edge embedding as $\mathbf{E}_{(i,j)} = \mathbf{r}$. However, due to fact sparsity, m tends to be small, which means many node pairs will not have labeled relations. Meanwhile, despite PLMs' limitations in common sense, they have shown some ability to encode commonsense knowledge (Davison et al., 2019; Petroni et al., 2019) and aid KG completion (Malaviya et al., 2019; Bosselut et al., 2019). Hence, we generate edge embeddings for the $\bar{m}$ unlabeled node pairs (*i.e.*, incomplete

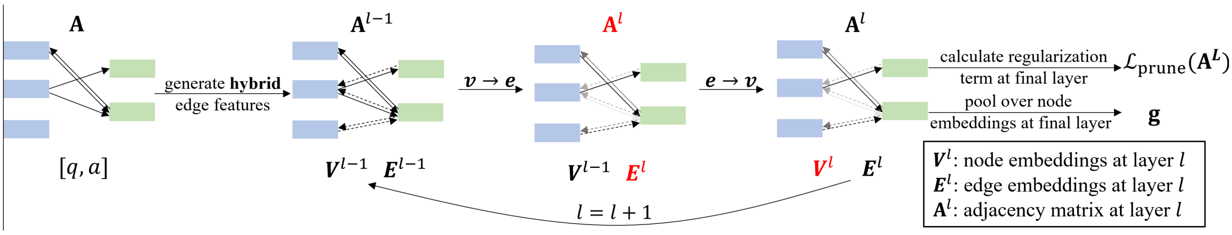

A
$\mathbf{A}^{l-1}$
$\mathbf{A}^{l}$
$\mathbf{A}^{l}$
generate **hybrid** edge features
$v \rightarrow e$
$e \rightarrow v$
calculate regularization term at final layer
$\mathcal{L}_{\text{prune}}(\mathbf{A}^L)$
pool over node embeddings at final layer
g
$[q, a]$
V^{l-1} E^{l-1}
V^{l-1} E^{l}
V^{l} E^{l}
$l = l + 1$
V^l: node embeddings at layer l
E^l: edge embeddings at layer l
$\mathbf{A}^l$: adjacency matrix at layer l

Figure 3: **Overview of our HGN's graph module. We jointly learn the graph structure and network parameters.** Darkness of edges indicate their weights. Red variables are updated in the previous step.

facts) by feeding each of these node pairs into a PLM-based generator $f_{\text{gen}}(\cdot, \cdot)$.

Edge Generation Inspired by Bosselut et al. (2019), we frame edge generation as text generation. For each extracted fact $(h, r, t) \in \mathcal{F}$, we first tokenize its node pair (h, t) and relation r. Let $\tilde{h}$, $\tilde{r}$, and $\tilde{t}$ be the respective token sequences of h, r, and t. Also, let \$ be a special separator token. For each tokenized extracted fact, we train the PLM to autoregressively generate the concatenated sequence $[\tilde{h}; \$; \tilde{t}; \$; \tilde{h}; \tilde{r}; \tilde{t}]$. During inference, we only have incomplete facts of the form $e = (h, t)$, with no r. Thus, for each e, the PLM is given $s_{\text{input}} = [\tilde{h}; \$; \tilde{t}; \$]$ and asked to generate the missing tokens $s_{\text{target}} = [\tilde{h}; \tilde{r}; \tilde{t}]$. Let s_{pred} be the PLM's prediction of s_{target}, and let $[x_1, x_2, ..., x_T] = [s, s_{\text{pred}}]$. The edge embedding for e is then computed as $f_{\text{gen}}(h, t) = \frac{1}{T} \sum_{i=1}^{T} \mathbf{h}_i$, where $\mathbf{h}_i$ is the PLM hidden state for x_i. See Appendix A for more details. Alternatively, we also consider the edge generation approach proposed in Wang et al. (2020), where $f_{\text{gen}}(\cdot, \cdot)$ is trained to generate a relational path connecting h and t. Such paths have been shown to contain useful semantic information for inferring the relation between h and t (Neelakantan et al., 2015; Das et al., 2017).

Hybrid Edge Embeddings In summary, edge embedding $\mathbf{E}_{(i,j)}$ for node pair $(v_i, v_j) \in \mathcal{V}'_{qa}$ is computed in a hybrid way: **(1)** If there exists $r \in \mathcal{R}$ such that $(v_i, r, v_j) \in \mathcal{E}'$, then $\mathbf{E}_{(i,j)} = \mathbf{r} \in R$. **(2)** Otherwise, $\mathbf{E}_{(i,j)} = f_{\text{adapt}}(f_{\text{gen}}(v_i, v_j))$, where $f_{\text{adapt}}(\cdot)$ is a MLP used to transform $f_{\text{gen}}(v_i, v_j)$ into the same space as $\mathbf{r}$.

3.4 Hybrid Subgraph Reasoning

The procedure described in §3.3 yields a hybrid subgraph, containing unweighted edges between all question-answer node pairs. However, some edges may not be relevant for answering the question, so HGN is designed to softly prune irrelevant edges. This is done by converting the unweighted subgraph into a weighted subgraph, then iteratively reweighting all edges while reasoning over the subgraph.

Following the Graph Network (GN) formulation proposed in (Battaglia et al., 2018), HGN's graph reasoning module consists of layer-wise node-to-edge ($v \rightarrow e$) and edge-to-node ($e \rightarrow v$) message passing functions. However, we equip HGN with a modified version of GN's edge-to-node message passing function, in which each edge's weight is used to rescale message flow on that edge. Intuitively, an edge's weight signifies the edge's relevance or helpfulness for reasoning about the given task instance.

Node-to-Edge Message Passing In node-to-edge message passing, HGN updates the edge embedding and edge weight of each question-answer node pair $(v_i, v_j) \in \mathcal{V}'_{qa}$, with respect to the (q, a) text encoding $\mathbf{s}$. Let $\mathbf{h}^{\ell}_{(i,j)}$ be the updated edge embedding for node pair $(v_i, v_j) \in \mathcal{V}'_{qa}$ at layer ℓ, where $\mathbf{h}^{0}_{(i,j)} = \mathbf{E}_{(i,j)}$. Let $\mathbf{A}^{\ell}_{(i,j)}$ be the updated edge weight for node pair $(v_i, v_j) \in \mathcal{V}'_{qa}$ at layer ℓ, where $\mathbf{A}^{0}_{(i,j)} = \mathbf{A}_{(i,j)}$. Since edge weights are normalized as $\mathbf{A}^{\ell}_{(i,j)} \in [0, 1]$, edges judged as irrelevant by HGN are softly pruned by being assigned near-zero weight.

Edge-to-Node Message Passing In edge-to-node message passing, HGN updates the embedding of each node $v_j \in \mathcal{V}'$, with respect to its neighbors' edge embeddings. Let N_j be the set of v_j's incoming neighbors. For each incoming neighbor $v_i \in N_j$, edge weight $\mathbf{A}^{\ell}_{(i,j)}$ is used to rescale the influence of v_i on v_j's embedding update. Let $\mathbf{h}^{\ell}_{j}$ be the updated node embedding for node v_j at layer ℓ, where $\mathbf{h}^{0}_{j} = \mathbf{v}_j$.

Update Rule Let $f^{\ell}_{v \rightarrow e}$, f^{ℓ}_{w}, f^{ℓ}_{u} and $f^{\ell}_{e \rightarrow v}$ be message passing MLPs. Formally, HGN's update rule is defined as:

$$
\begin{aligned}
v \rightarrow e : \mathbf{h}^{\ell}_{(i,j)} &= f^{l}_{v \rightarrow e}\left(\left[\mathbf{h}^{\ell-1}_{i}; \mathbf{h}^{\ell-1}_{j}; \mathbf{h}^{\ell-1}_{(i,j)}; \mathbf{s}\right]\right); \\
w^{\ell}_{(i,j)} &= f^{\ell}_{w}\left(\left[\mathbf{h}^{\ell-1}_{(i,j)}; \mathbf{s}\right]\right); \\
\mathbf{A}^{\ell}_{(i,j)} &= \frac{e^{w^{\ell}_{(i,j)}}}{\sum_{(s,t) \in E} e^{w^{\ell}_{(s,t)}}}, \\
e \rightarrow v : \mathbf{u}^{\ell}_{(i,j)} &= f^{\ell}_{u}\left(\left[\mathbf{h}^{\ell-1}_{i}; \mathbf{h}^{\ell}_{(i,j)}\right]\right); \\
\mathbf{h}^{\ell}_{j} &= f^{\ell}_{e \rightarrow v}\left(\sum_{v_i \in \mathcal{N}_j} \mathbf{A}^{\ell}_{(i,j)} \mathbf{u}^{\ell}_{(i,j)}\right).
\end{aligned}
\tag{1}
$$

Globally Normalized Edge Weights Whereas Graph Attention Network (GAT) (Veličković et al., 2017) normalizes (v_i, v_j)'s edge weight over all edges in v_j's neighborhood (local normalization), HGN normalizes over all edges in the subgraph (global normalization). We use global normalization because local normalization assumes at least one edge in v_j's neighborhood is relevant, which may not be true. For example, given an irrelevant or incorrectly grounded concept, none of its edges will be helpful, and so all edges in its neighborhood should be pruned. Such irrelevant nodes should be softly excluded from message passing, as opposed to letting them continue to influence the reasoning process. To show the benefit of global edge attention, we empirically compare our default HGN architecture to an HGN variant using GAT-style attention (§4.3).

Graph Encoding After L layers of message passing, we obtain node embeddings $\mathbf{V}^L$ and edge embeddings $\mathbf{E}^L$. $\mathbf{V}^L$ is aggregated into a single node embedding $\mathbf{v}_{\text{agg}}$ via attention pooling, and $\mathbf{E}^L$ is aggregated into a single edge embedding $\mathbf{e}_{\text{agg}}$ via edge-weighted sum pooling. The final graph encoding is then given as $\mathbf{g} = [\mathbf{v}_{\text{agg}}, \mathbf{e}_{\text{agg}}]$.

3.5 Learning Objective

HGN's learning objective is $\mathcal{L} = \mathcal{L}_{\text{task}} + \beta \mathcal{L}_{\text{prune}}$, where $\mathcal{L}_{\text{task}}$ is the cross-entropy loss for the QA task, $\mathcal{L}_{\text{prune}}$ is our proposed entropy regularization of $\mathbf{A}^L$, and $\beta \in [0, \infty)$ is a hyperparameter for weighting $\mathcal{L}_{\text{prune}}$. We train HGN end-to-end using the RAdam (Liu et al., 2020) optimizer. We explain $\mathcal{L}_{\text{prune}}$ in the following paragraph.

Entropy Regularization for Edge Pruning To encourage the model to take decisive pruning steps when refining the subgraph structure, we use a regularization term $\mathcal{L}_{\text{prune}}$ to penalize non-discriminative edge weights. In an extreme case, a blind model will assign the same weight to all edges, degenerating G into an unweighted graph. This is a failure case, since hybrid subgraphs are likely to contain mostly irrelevant edges, and we want the model to focus on the relevant edges. Therefore, via $\mathcal{L}_{\text{prune}}$, we train the model to minimize the entropy of the edge weight distribution (*i.e.,* make the distribution more skewed), in order to maximize the informativeness of the predicted edge weights. Lower entropy means the model has higher certainty about edges' relevance to the given task instance, such that the model will discriminatively judge some edges as being much more relevant than others. $\mathcal{L}_{\text{prune}}$ is calculated as:

$$
\begin{aligned}
&\mathcal{L}_{\text{prune}}(\mathbf{A}^L(q, a; \theta_{\text{text}}, \theta_{\text{graph}})) \\
&= - \sum_{(i,j):(v_i,v_j) \in \mathcal{V}'_{qa}} \mathbf{A}^L_{(i,j)} \log \mathbf{A}^L_{(i,j)}.
\end{aligned}
\tag{2}
$$

4 Experiments

4.1 Experiment Setup

We evaluate our proposed model on four multiple-choice commonsense QA datasets: **CommonsenseQA** (Talmor et al., 2019), **CODAH** (Chen et al., 2019), **OpenbookQA** (Mihaylov et al., 2018) and **QASC** (Khot et al., 2020) (details in §B). We use ConceptNet (Speer et al., 2017), a commonsense knowledge graph, as $\mathcal{G}$. For text encoder f_{text}, we experiment with BERT-Base, BERT-Large (Devlin et al., 2019) and RoBERTa(-Large) (Liu et al., 2019) to validate our model's effectiveness over different text encoders. We use GPT-2 (Radford et al., 2019) for f_{gen}. For OpenbookQA and QASC, retrieving related facts from the provided corpus plays an important role in boosting the model's performance. Therefore, we build our graph reasoning model on top of retrieval-augmented methods on the leaderboard: "AristoRoBERTa"[2] for OpenbookQA and "RoBERTa (2-step IR)"[3] for QASC. In this way, we can study if strong retrieval-augmented methods could still benefit from KG knowledge and our reasoning framework.

[2] https://leaderboard.allenai.org/open_book_qa/submission/blcp1tu91i4gm0vf484g

[3] https://leaderboard.allenai.org/qasc/submission/bolaun0ghifmkohgvhr0

Methods	BERT-Base		BERT-Large		RoBERTa	
	60% Train	100% Train	60% Train	100% Train	60% Train	100% Train
LM Finetuning	52.06 (±0.72)	53.47 (±0.87)	52.30 (±0.16)	55.39 (±0.40)	65.56 (±0.76)	68.69 (±0.56)
RN (Santoro et al., 2017)	54.43 (±0.10)	56.20 (±0.45)	54.23 (±0.28)	58.46 (±0.71)	66.16 (±0.28)	70.08 (±0.21)
RN + Link Prediction	-	-	53.96 (±0.56)	56.02 (±0.55)	66.29(±0.29)	69.33 (±0.98)
RGCN (Schlichtkrull et al., 2018b)	52.20 (±0.31)	54.50 (±0.56)	54.71 (±0.37)	57.13 (±0.36)	68.33 (±0.85)	68.41 (±0.66)
GAT (Veličković et al., 2017)	53.05 (±0.37)	56.51 (±0.74)	55.80 (±0.53)	58.18 (±1.07)	69.63 (±0.42)	71.20 (±0.72)
GN (Battaglia et al., 2018)	53.67 (±0.45)	55.65 (±0.51)	54.78 (±0.61)	57.81 (±0.67)	68.78 (±0.67)	71.12 (±0.45)
GconAttn (Wang et al., 2019a)	51.36 (±0.98)	54.41 (±0.50)	54.96 (±0.69)	56.94 (±0.77)	68.09 (±0.63)	69.88 (±0.47)
KagNet (Lin et al., 2019)	-	56.19	-	57.16	-	-
MHGRN (Feng et al., 2020)	54.12 (±0.49)	56.23 (±0.82)	56.76 (±0.21)	59.85 (±0.56)	68.84 (±1.06)	71.11 (±0.81)
PathGenerator (Wang et al., 2020)	54.44 (±0.42)	56.99 (±0.41)	57.53 (±0.19)	59.07 (±0.30)	69.46 (±0.23)	72.68 (±0.42)
HGN	**55.72** (±0.32)*	**58.01** (±0.29)*	**57.78** (±0.41)†	**61.11** (±0.21)*	**71.10** (±0.11)*	**73.64** (±0.30)*
HGN (w/ PathGen evidence)	55.68 (±0.29)*	57.77 (±0.39)*	57.63 (±0.25)†	60.89 (±0.19)*	70.95 (±0.21)*	73.41 (±0.31)*

Table 1: **Accuracy on CommonsenseQA inhouse test set.** We use the same inhouse split as Lin et al. (2019). Some baseline results are reported by Feng et al. (2020) and Wang et al. (2020). Mean and standard deviation of four runs are presented for all models except KagNet. $*$ indicates significant improvement over all baselines and † indicates significant improvement over all baselines except PathGenerator.

4.2 Compared Methods

We compare our model with a series of KG-augmented methods:

Models Using Extracted Facts: RN (Santoro et al., 2017) builds the graph with the same node set as our method but extracted edges only. The graph vector is calculated as $g = \text{Pool}(\{\text{MLP}([\mathbf{x}_i; \mathbf{x}_{(i,j)}; \mathbf{x}_j]) \mid (v_i, r, v_j) \in \mathcal{F}\})$. **GN** (Battaglia et al., 2018) presents a general formulation of GNNs. We instantiate it with the layerwise propagation rule defined in Eq. 1. It differs from our HGN in that: (1) it only considers extracted edges; (2) all edge weights are fixed to 1. **MHGRN** (Feng et al., 2020) generalizes GNNs with multi-hop message passing. **GAT** (Veličković et al., 2017) adopts attention mechanism to reweight edges locally in each node's neighborhood. To get the graph vector, we use the same pooling function as HGN. Descriptions for **RGCN**, **GconAttn**, and **KagNet** can be found in §D.

Models Using Generated Facts: RN + Link Prediction differs from RN by only considering the generated relation (predicted using TransE (Bordes et al., 2013)) between question and answer concepts. **PathGenerator** (Wang et al., 2020) learns a path generator from paths collected through random walks on the KG. The learned generator is used to generate paths connecting question and answer concepts. Attentive pooling is used to derive the graph vector given a set of path embeddings.

Our Model's Variant: HGN (w/ PathGen evidence) As described in §3.3.2, we consider a variant of HGN which uses PathGenerator (PathGen) (Wang et al., 2020) as f_{gen} to generate edge embeddings for unlabeled node pairs.

Methods	BERT-Large	RoBERTa
LM Finetuning	65.74	83.14
RN (Santoro et al., 2017)	64.59	82.45
RGCN (Schlichtkrull et al., 2018b)	65.56	82.42
GAT (Veličković et al., 2017)	65.88	82.78
GconAttn (Wang et al., 2019a)	65.17	82.35
GN (Battaglia et al., 2018)	65.52	82.06
MHGRN (Feng et al., 2020)	65.92	83.07
PathGenerator (Wang et al., 2020)	64.67	82.27
HGN	**66.75**	84.08
HGN (w/ PathGen evidence)	65.96	**84.32**

Table 2: **Test accuracy on CODAH.** We perform 5-fold cross validation with the official split.

Datasets Base Models	OpenbookQA AristoRoBERTa	QASC RoBERTa (2-step IR)
LM Finetuning	77.40 (±1.64)	73.34 (±0.71)
RN (Santoro et al., 2017)	75.35 (±1.39)	72.77 (±1.50)
RN + Link Prediction	77.25 (±1.11)	-
RGCN (Schlichtkrull et al., 2018b)	74.60 (±2.53)	72.23 (±1.36)
GAT (Veličković et al., 2017)	78.20 (±1.22)	72.61 (±0.93)
GN (Battaglia et al., 2018)	77.25 (±0.91)	72.53 (±0.70)
GconAttn (Wang et al., 2019a)	71.80 (±1.21)	72.72 (±1.66)
MHGRN (Feng et al., 2020)	77.75 (±0.38)	73.24 (±0.45)
PathGenerator (Wang et al., 2020)	79.15 (±0.78)	72.96 (±0.68)
HGN	**80.15** (±0.38)*	**74.27** (±0.31)†
HGN (w/ PathGen evidence)	80.05 (±0.54)†	74.00 (±1.31)

Table 3: **Test accuracy on OpenbookQA and QASC with retrieval-augmented methods as base model.** Some baseline results are reported by Feng et al. (2020) and Wang et al. (2020). Mean and standard deviation of four runs are presented for all models.

4.3 Results

Performance Comparisons. Tables 1, 2, 3 show performance comparisons between our models and baseline models on CommonsenseQA, OpenbookQA, CODAH and QASC. Our HGN shows

Methods	Text Encoder	Test Acc
UnifiedQA (Khashabi et al., 2020)	T5-11B	87.2
T5-11B + KB	T5-11B	85.4
T5-3B (Raffel et al., 2020)	T5-3B	83.2
PathGenerator (Wang et al., 2020)	ALBERT	81.8
HGN (ours)	**AristoRoBERTa**	**81.4**
AristoRoBERTa + KB	AristoRoBERTa	81.0
MHGRN (Feng et al., 2020)	AristoRoBERTa	80.6
PathGenerator (Wang et al., 2020)	AristoRoBERTa	80.2
KF + SIR (Banerjee and Baral, 2020)	RoBERTa	80.2
AristoRoBERTa	AristoRoBERTa	80.2

Table 4: **Leaderboard of OpenbookQA.** Our HGN ranks first among all submissions using AristoRoBERTa as the text encoder.

Methods	Single	Ensemble
ALBERT+DESC-KCR (Xu et al., 2020)	80.7	83.3
ALBERT+KD	80.3	80.9
ALBERT+KCR	79.5	-
Unified QA (Khashabi et al., 2020)	79.1	-
ALBERT+KRD	78.4	-
T5-3B (Raffel et al., 2020)	78.1	-
Albert+HGN (ours)	**77.3**	**80.0**
TeGBERT	76.8	-
ALBERT+PathGenerator (Wang et al., 2020)	75.6	78.2
ALBERT (Lan et al., 2019)	-	76.5

Table 5: **Leaderboard of CommonsenseQA.** Our HGN gets remarkable improvement over PathGenerator (Wang et al., 2020).

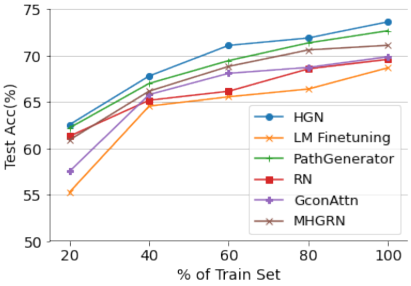


Figure 4: **Performance with different amounts of training data for CommonsenseQA (RoBERTa).**

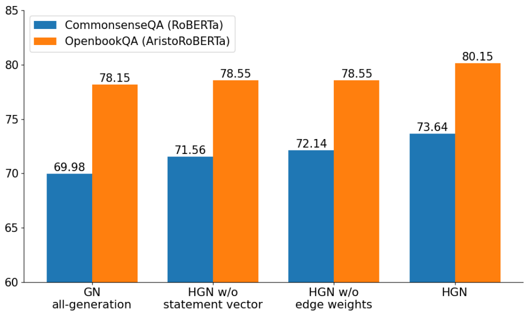


Figure 5: **Performance of different model variants.**

consistent improvement over baseline models on all datasets. The improvement over all baselines are tested to be statistically significant under most settings, demonstrating the effectiveness of HGN.

We also submit our best model to leaderboards for CommonsenseQA and OpenbookQA. For CommonsenseQA (Table 5), UnifiedQA (Khashabi et al., 2020) and T5-3B (Raffel et al., 2020) have 11B and 3B hyperparameters respectively, making them impractical to be finetuned in an academic setting. ALBERT+DESC-KCR (Xu et al., 2020), ALBERT+KD additionally use concept definitions from dictionaries and ALBERT+KRD retrieve sentences from OMCS corpus (Liu and Singh, 2004) as input and thus not comparable with our model. ALBERT+KCR incorporates ConceptNet triples into the input sequence to enhance the text encoder and the advantage should be orthogonal to that brought by improving the graph encoder. Our HGN shows remarkable improvement over PathGenerator (Wang et al., 2020) and the LM Finetuning approach (ALBERT (Lan et al., 2019)). For OpenbookQA (Table 4), our model ranks the first among all models using AristoRoBERTa as the text encoder.

Training with Less Labeled Data. Fig. 4 shows the results of our model and baseline models when trained with different portions of the training data on CommonsenseQA. Our model gets better test accuracy under all settings. The improvement over the knowledge-agnostic baseline (LM Finetuning) is more significant with less training data, which suggests that incorporating external knowledge is more helpful in the low-resource setting.

Study on More Model Variants. To better understand the model design, we experiment with three variants on CommonsenseQA and OpenbookQA. **GN all-generation** doesn't consider extracted facts and instead generate edge features between all question and answer concepts. **HGN w/o statement vector** doesn't consider **s** in Eq. 1, which isolates the graph encoder from the text encoder. **HGN w/o edge weights** reasons over an unweighted graph with hybrid features, which means edge weights are all fixed to 1 during training. Fig. 5 shows the results of the ablation study. Comparing "GN all-generation" with "HGN w/o edge weights", we can conclude that extracted facts play an important role in HGN and can't be replaced by generated features. The high precision of extracted facts is still desirable even if we have a model to generate relational edges. Comparing "HGN w/o statement vector" with "HGN", we find that accessing context information is also important for graph reasoning, which means information propagation and edge weight prediction should be conducted in a context-

Contextualized Graph	**GN (A)**	**HGN ($\mathbf{A}^K$)**
Number of Edges	3.65 (±2.73)	4.38 (±3.24)
Number of Valid Edges	2.67 (±1.95)	3.15 (±1.98)
Percentage of Valid Edges	71.64%	78.51%
Average Helpfulness Score of Edges	0.90 (±0.50)	1.16 (±0.51)
Prune Rate	-	77.13%

Table 6: **User studies on learned graph structures.** 30 pairs of contextualized graphs output by GN and HGN are evaluated by 5 annotators.

aware manner. HGN also improves over "HGN (w/o edge weights)", indicating the effectiveness of conducting context-dependent pruning.

4.4 User Studies on Learned Graph Structures

To assess our model's ability to refine the graph structure, we compare the graph structure before and after being processed by HGN. Specifically, we sample 30 questions with its correct answer from the development set of CommonsenseQA and ask 5 human annotators to evaluate the graph output by GN (with adjacency matrix $\mathbf{A}$ and extracted facts only) and our HGN (with adjacency matrix $\mathbf{A}^L$). We manually binarizing $\mathbf{A}^L$ by removing edges with weight lower than 0.01.

Given a graph, for each edge (fact), annotators are asked to rate its **validness** and **helpfulness**. The validness score is rated as a binary value in a context-agnostic way: 0 (the fact doesn't make sense), 1 (the fact is generally true). The helpfulness score measures if the fact is helpful for solving the question and is rated on a 0 to 2 scale: 0 (the fact is unrelated to the question and answer), 1 (the fact is related but doesn't directly lead to the answer), 2 (the fact directly leads to the answer). The mean ratings for 30 pairs of (GN, HGN) graphs by 5 annotators are reported in Table 6. We also include another metric named "prune rate" calculated as: $1 - \frac{\text{\# edges in binarized } \mathbf{A}^L}{\text{\# edges in } \mathbf{A}^0}$, which measures the portion of edges that are assigned very low weights (softly pruned) during training and is only applicable to HGN. The Fleiss' Kappa (Fleiss, 1971) is 0.51 (moderate agreement) for validness and 0.36 (fair agreement) for helpfulness. The graph refined by HGN has both more edges and more valid edges compared to the extracted one. The refined graph also achieves a higher helpfulness score. These all indicate that our HGN learns a superior graph structure with more helpful edges and less noisy edges, which is the reason for performance improvement over previous works that rely on extracted and static graphs. Detailed cases can be found in §C.

5 Related Work

Commonsense Question Answering. Answering commonsense questions is challenging because the required commonsense knowledge is neither written down in the context nor held by pretrained language models. Therefore, many works leverage external knowledge to obtain additional evidence. These works can be categorized into IR-augmented methods, where evidence is retrieved from text corpora, and KG-augmented methods, where evidence is collected from KGs. Lv et al. (2020) demonstrate that IR-based evidence and KG-based evidence are complementary to each other. Although adding IR evidence can lead to further performance improvement, we only focus on the challenges of KG-augmented methods in this paper. Literature in this domain mainly studies how to encode the contextualized subgraph extracted from a KG. For example, Lin et al. (2019) propose a model comprised of GCN and LSTM to account for both the global graph structure and local paths connecting question concepts and answer concepts. Ma et al. (2019) use BERT to generate the embedding for the pseudo-sentence representing each edge and then adopt the attention mechanism to aggregate edge features as the graph encoding. Crucial difference is that they assume a static graph and there's no operation on enriching or denoising the graph structure. While Wang et al. (2020) also complete the contextualized graph with a path generator, they still reason over the static graph and neglect the noise introduced during generation.

Graph Structure Learning. Works that jointly learn the graph structure with the downstream task can be classified into two categories. One line of works directly learn an unweighted graph with desired edges for reasoning. Kipf et al. (2018) and Franceschi et al. (2019) sample the graph structure from a predicted probabilistic distribution with differentiable approximations. Norcliffe-Brown et al. (2018) calculate the relatedness between any pair of nodes and only keep the top-k strongest connections for each node to construct the edge set. Sun et al. (2019) start with a small graph and iteratively expand it with a set of retrieving operations. The other line of works consider a weighted graph with all possible edges and softly filter out the noisy ones by downweighting them. An adjacency matrix with continuous values is incorporated into message passing. Jiang et al. (2019) and Yu et al. (2019) use heuristics to regularize the learned adjacency matrix. Hu et al. (2019) consider the question embedding for predicting edge weights. Our HGN falls into the second category and therefore avoids information loss caused by hard pruning and approximation. Our uniqueness is that we construct the graph with hybrid features based on extracted and generated facts and we let node features, edge features, edge weights, and the global signal (statement vector) collectively determine the evolution of the graph structure. These empower our model with greater capacity and flexibility for KG-augmented QA.

6 Conclusion

In this paper, we propose a neural-symbolic framework for commonsense reasoning named HGN. To address the issues with missing facts from external knowledge graph and noisy facts from the contextualized knowledge graph, our proposed HGN jointly generates features for new edges, refines the graph structure, and learns the parameters for graph networks. Experimental results and user studies demonstrate the effectiveness of our model. In the future, we plan to incorporate open relations in graph initialization, which are more expressive than predefined KG relations. We also plan to study how to make the fact generation or extraction process aware of the reasoning context.

References

Ian Apperly. 2010. *Mindreaders: the cognitive basis of" theory of mind"*. Psychology Press.

Pratyay Banerjee and Chitta Baral. 2020. Knowledge fusion and semantic knowledge ranking for open domain question answering. *arXiv preprint arXiv:2004.03101*.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

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

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.

Michael Chen, Mike D'Arcy, Alisa Liu, Jared Fernandez, and Doug Downey. 2019. Codah: An adversarially-authored question answering dataset for common sense. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 63–69.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 132–141.

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.

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.

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*.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

L Franceschi, M Niepert, M Pontil, and X He. 2019. Learning discrete structures for graph neural networks. In *Proceedings of ICML*, volume 97. PMLR.

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

Ronghang Hu, Anna Rohrbach, Trevor Darrell, and Kate Saenko. 2019. Language-conditioned graph networks for relational reasoning. In *Proceedings of CVPR*, pages 10294–10303.

Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. 2019. Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of CVPR*, pages 11313–11320.

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

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *AAAI*, pages 8082–8090.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. 2018. Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

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.

Hugo Liu and Push Singh. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the variance of the adaptive learning rate and beyond. In *Proceedings of ICLR*.

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*.

Shangwen Lv, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. 2020. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering. In *Proceedings of AAAI*, pages 8449–8456.

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.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2019. Exploiting structural and semantic context for commonsense knowledge base completion. *arXiv preprint arXiv:1910.02915*.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. In *Proceedings of AAAI*, pages 2925–2933.

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 EMNLP*, pages 2381–2391.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 156–166.

Will Norcliffe-Brown, Stathis Vafeias, and Sarah Parisot. 2018. Learning conditioned graph structures for interpretable visual question answering. In *Advances in neural information processing systems*, pages 8334–8343.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

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.

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

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018b. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.

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.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of EMNLP-IJCNLP*, pages 2380–2390.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of NAACL-HLT*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

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, and Michael Witbrock. 2019a. Improving natural language inference using external knowledge in the science questions domain. In *Proceedings of AAAI*, pages 7208–7215. AAAI Press.

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

Yichong Xu, Chenguang Zhu, Ruochen Xu, Yang Liu, Michael Zeng, and Xuedong Huang. 2020. Fusing context into knowledge graph for commonsense reasoning. *arXiv preprint arXiv:2012.04808*.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. G-daug: Generative data augmentation for commonsense reasoning. *arXiv preprint arXiv:2004.11546*.

Donghan Yu, Ruohong Zhang, Zhengbao Jiang, Yuexin Wu, and Yiming Yang. 2019. Graph-revised convolutional network. In *Proceedings of ECML-PKDD*.

A Implementation Details of Edge Feature Generator

As an implementation of f_{gen}, we adopt GPT-2 (Radford et al., 2019), which is pretrained on large corpora and achieves great success on a wide range of tasks involving sentence generation, as a generator to generalize the facts from the knowledge graph. We first convert each fact $(h, r, t) \in \mathcal{F}$ into a word sequence with a "prompt-generation" format: $\left[\tilde{h}, \$, \tilde{t}, \$, \tilde{h}, \tilde{r}, \tilde{t}\right]$, where $\tilde{h}, \tilde{r}, \tilde{t}$ are the word sequence of h, r, t respectively, \$ denotes the delimiter token used by GPT-2, and $[\cdot;\cdot]$ denotes word sequence concatenation. We denote the synthetic sentence as $s_{(h,r,t)} = \left[x_1^{(h,r,t)}, \ldots, x_{n_{(h,r,t)}}^{(h,r,t)}\right]$ and finetune GPT-2 on all synthetic sentences created from $\mathcal{F}$ with the language modeling objective:

$$\mathcal{L}_{\text{gen}}(\mathcal{F}) = \sum_{(h,r,t)\in\mathcal{F}} \sum_{i=1}^{n_{(h,r,t)}} \log P\left(x_i^{(h,r,t)} \mid x_1^{(h,r,t)}, \ldots, x_{i-1}^{(h,r,t)}\right) \tag{3}$$

After that, given any two concepts (v_i, v_j), we build a prompt as $[\tilde{v_i}; \$; \tilde{v_j}; \$]$ and let the model to generate the following word sequence. We denote the whole sentence (both prompt and generation) as $s_{(v_i,v_j)}$, and the hidden states of each word during generation as $\mathbf{h}_1, \ldots, \mathbf{h}_T$ where T is the sentence length. We average hidden states of all words in the sentence to get the relational feature: $f_{\text{gen}}(v_i, v_j) = \frac{1}{T}\sum_{i=1}^{T} \mathbf{h}_i$.

B Details of Datasets

CommonsenseQA (Talmor et al., 2019) is a multiple-choice QA dataset targeting commonsense. It's constructed based on the knowledge in ConceptNet. Since the test set of the official split (9741/1221/1140 for OFtrain/OFdev/OFtest) is not publicly available, we compare our models with baseline models on the inhouse split (8500/1221/1241 for IHtrain/IHdev/IHtest)[4] used by previous works (Lin et al., 2019; Feng et al., 2020; Wang et al., 2020).

CODAH (Chen et al., 2019) contains 2801 sentence completion questions testing commonsense reasoning skills. We perform 5-fold cross validation following the practice in Yang et al. (2020). The authors shared their splits with us.

OpenbookQA (Mihaylov et al., 2018) is a multiple-choice QA dataset modeled after openbook exams. Besides 5957 elementary-level science questions (4957/500/500 for train/dev/test), it also provides an open book with 1326 core science facts. Solving the dataset requires combining facts from open book with commonsense knowledge.

QASC (Khot et al., 2020) is a QA dataset with questions about grade-school science. It has 9980 8-way multiple-choice questions (8134/926/920 train/dev/test), and comes with a corpus of 17M sentences. Since the official test set does not have labels, we create an in-house test split by randomly moving 920 questions from the train set to the test set. Solving questions in QASC requires retrieving facts from the corpus and composing them to produce an answer.

C Case Study

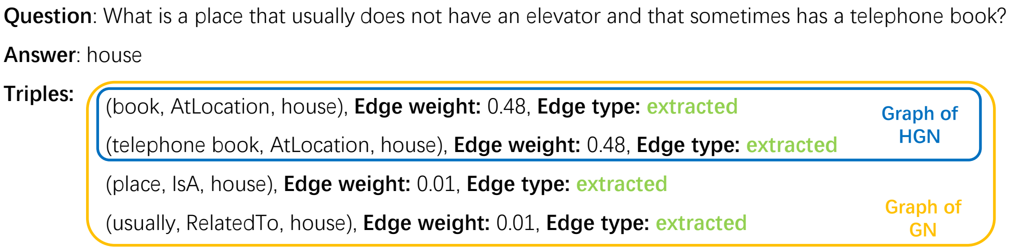
We compare the graph generated by our HGN with the extracted one (GN). On the development set, there are two dominating cases and we show the representative instance of each one. Figure 6 shows the first case, where HGN prunes edges from the extracted graph. Our HGN assigns the highest weights to the most helpful facts (`book, AtLocation, house`), (`telephone book, AtLocation, house`). It also downweight unhelpful fact (`place, IsA, house`) and invalid fact (`usually, RelatedTo, house`). Figure 7 shows the second case, where new generated facts are incorporated into reasoning. All generated facts that are kept by the model make sense in the context and help identify the answer. Both cases suggest that our model improve the quality of the contextualized knowledge graph compared to the current methods that only rely on extracted facts.

D Compared Methods

RGCN (Schlichtkrull et al., 2018a) extends Graph Convolutional Networks (GCNs) (Kipf and Welling, 2017) with relation-specific transition matrices during message passing. It operates on the same graph as RN. The graph vector is calculated as $\mathbf{g} = \text{Pool}(\{\mathbf{h}_i^K \mid v_i \in V\})$.

GconAttn (Wang et al., 2019b) softly aligns the nodes in question and answer and do pooling over

[4] https://github.com/INK-USC/MHGRN/blob/master/data/csqa/inhouse_split_qids.txt



Question: What is a place that usually does not have an elevator and that sometimes has a telephone book?

Answer: house

Triples:

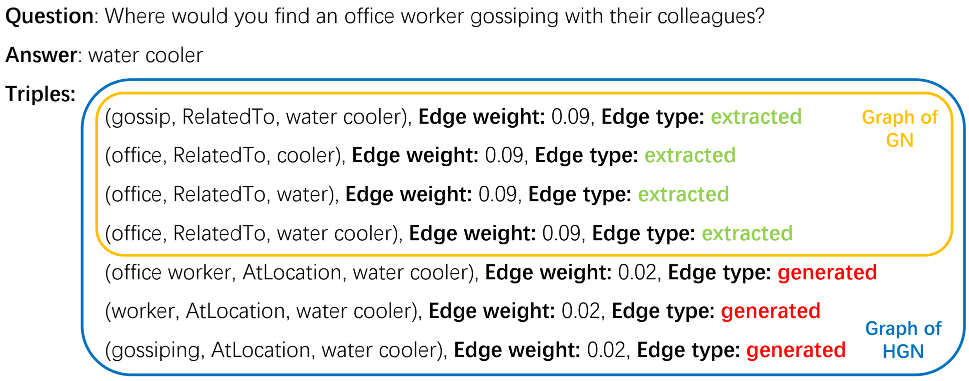
(book, AtLocation, house), **Edge weight:** 0.48, **Edge type:** extracted

(telephone book, AtLocation, house), **Edge weight:** 0.48, **Edge type:** extracted

(place, IsA, house), **Edge weight:** 0.01, **Edge type:** extracted

(usually, RelatedTo, house), **Edge weight:** 0.01, **Edge type:** extracted

Figure 6: **Case I: Unrelated extracted facts are filtered out.**



Question: Where would you find an office worker gossiping with their colleagues?

Answer: water cooler

Triples:

(gossip, RelatedTo, water cooler), **Edge weight:** 0.09, **Edge type:** extracted

(office, RelatedTo, cooler), **Edge weight:** 0.09, **Edge type:** extracted

(office, RelatedTo, water), **Edge weight:** 0.09, **Edge type:** extracted

(office, RelatedTo, water cooler), **Edge weight:** 0.09, **Edge type:** extracted

(office worker, AtLocation, water cooler), **Edge weight:** 0.02, **Edge type:** generated

(worker, AtLocation, water cooler), **Edge weight:** 0.02, **Edge type:** generated

(gossiping, AtLocation, water cooler), **Edge weight:** 0.02, **Edge type:** generated

Figure 7: **Case II: Helpful generated facts are incorporated.**

all matching nodes to get $\mathbf{g}$.

KagNet (Lin et al., 2019) uses an LSTM to encode relational paths between question and answer concepts and pool over the path embeddings for graph encoding.