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], IIT Delhi[2], Tsinghua University[3], Adobe Research[4]

Presented by Vardaan Pahuja and Akshay Kekuda

Commonsense Reasoning

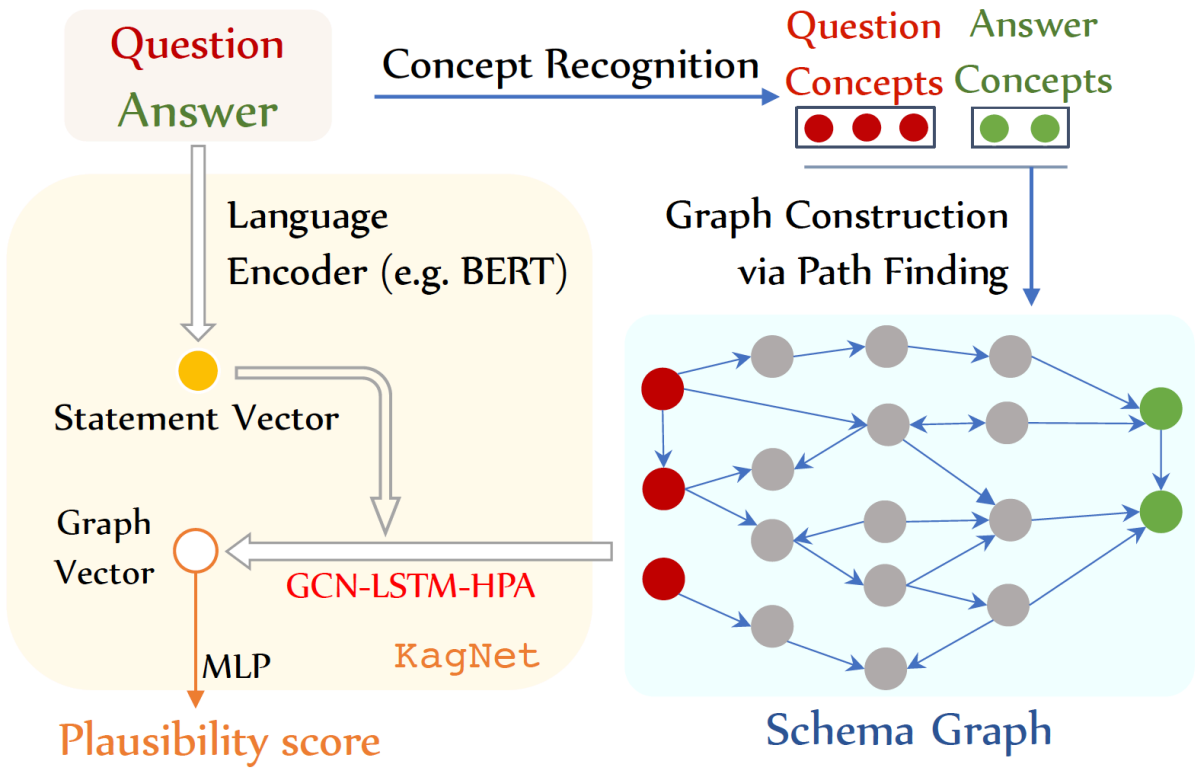
- Answer questions based on “common sense”
- Take into account real world knowledge, social conventions, scientific facts etc.

Knowledge Graphs for Commonsense

- **ConceptNet**
- Common-sense knowledge in the form of RDF triples
- Words and phrases of natural language are connected with labeled edges.
  - (River, atLocation, bridge)
  - (Dog, hasA, tail)
  - (Ice, hasProperty, cold)

KG augmented PLMs

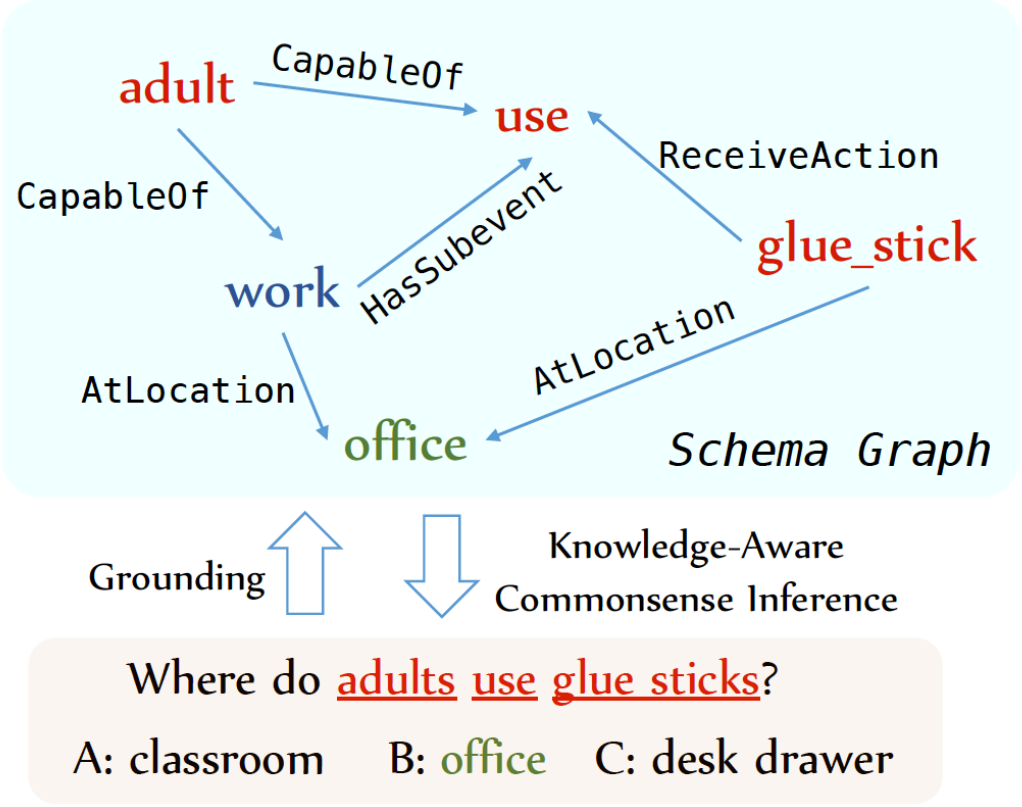
- Construct a context/schema graph including the question and answer nodes
- Use a GNN on schema graph for multi-hop reasoning



Courtesy: [Lin et al. 2019]

Challenge: Need useful KG facts

- Extraction
  - Find k-hop paths between question and answer concepts (KagNet, QA-GNN)



Courtesy: [Lin et al. 2019]

Challenge: Need useful KG facts

- Extraction
  - Find k-hop paths between question and answer concepts (KagNet, QA-GNN)
- Edge generation using PLMs
  - **COMET** [Bosselut et al. 2019]
  - Use transformer model (GPT) to generate new commonsense facts
  - **Training task**: Given the subject and relation tokens, the model generates the object tokens.

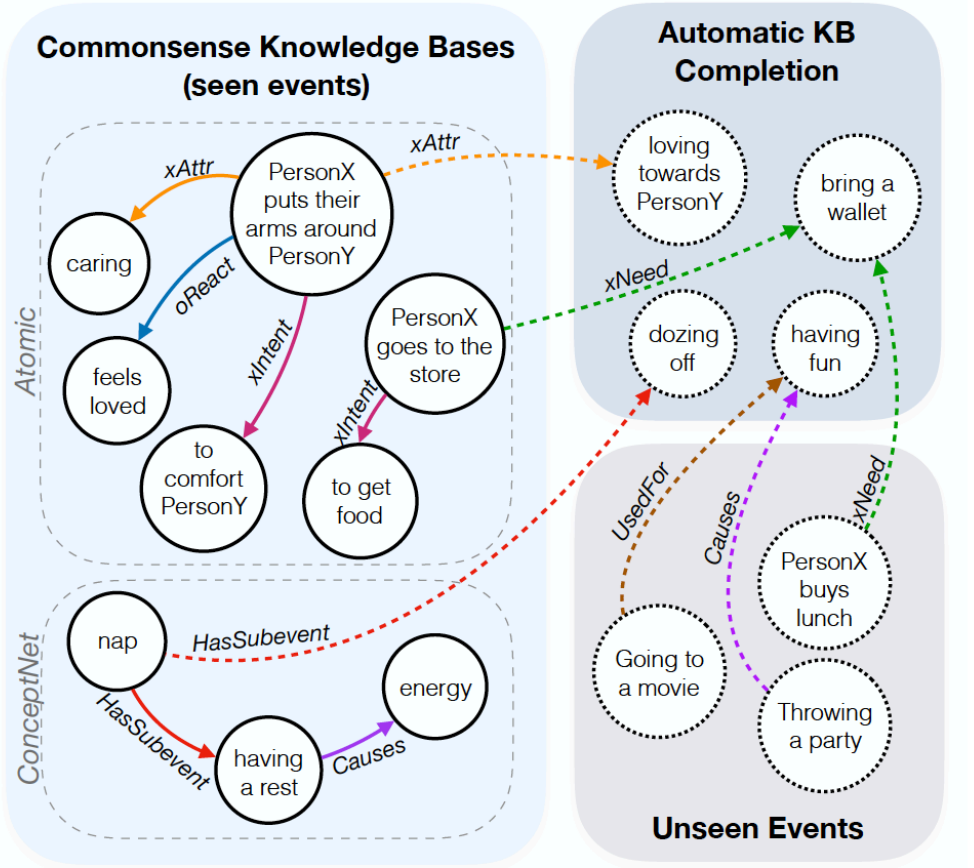


Figure 1: COMET learns from an existing knowledge base (solid lines) to be able to generate novel nodes and edges (dashed lines).

Challenge: Need useful KG facts

- Low Precision
  - Can result from bad annotations in KG
- Low Recall
  - Missing edges in KG
- Noisy extraction or generation of edges

Challenge: Need useful KG facts

- Low Precision
  - Can result from bad annotations in KG
- Low Recall
  - Missing edges in KG
- Noisy extraction or generation of edges
- Both extraction and generation may introduce unhelpful edges
- Need a way to filter/down-weight the importance of irrelevant edges

This Work

- Models the interactions b/w extracted and generated edges jointly
- Identify **contextually relevant** edges

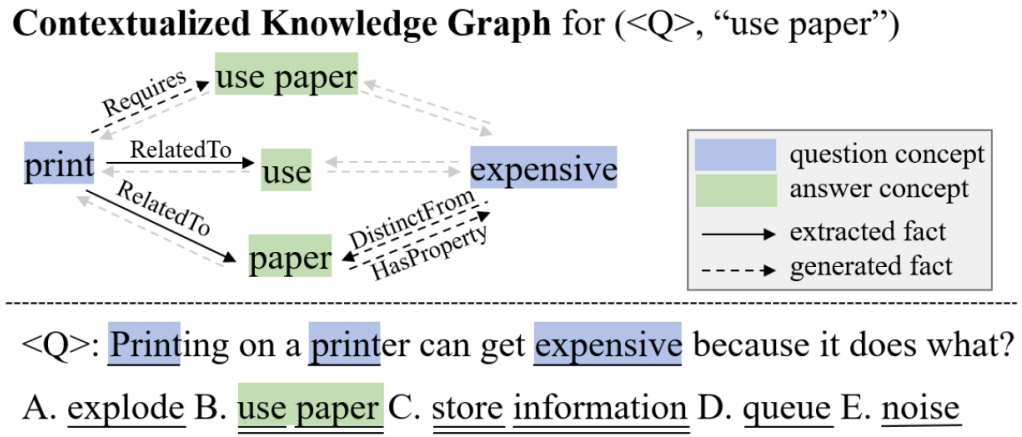


Figure 1: **KG-Augmented Commonsense QA.** Predicting the correct answer ("use paper") requires commonsense facts like `(print, Requires, paper)` and `(paper, HasProperty, expensive)`, which are not given in the question and candidate answers. HGN uses facts extracted from the KG, *e.g.*, `(print, RelatedTo,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

Hybrid Graph Construction

- Step1: **Concept Grounding**
  - Find relevant text spans in question and answer that match nodes in KG
  - Node features: BERT or TransE
- Step2: **Build graph**
  - Use fully-connected graph

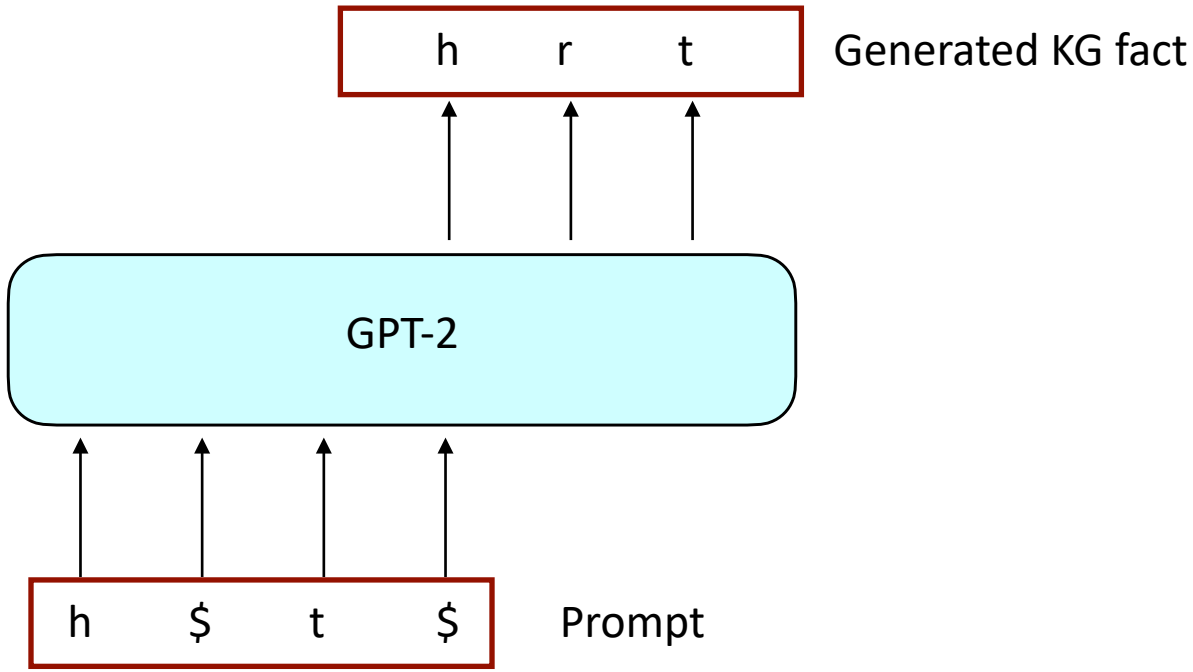
$$\mathcal{E}' = (\mathcal{V}'_q \times \mathcal{V}'_a) \cup (\mathcal{V}'_a \times \mathcal{V}'_q)$$

How to obtain edge features?

- For edges extracted from KG, initialize them using learned relation embeddings from KBE models.
- Many edges don’t have labeled relations in KG

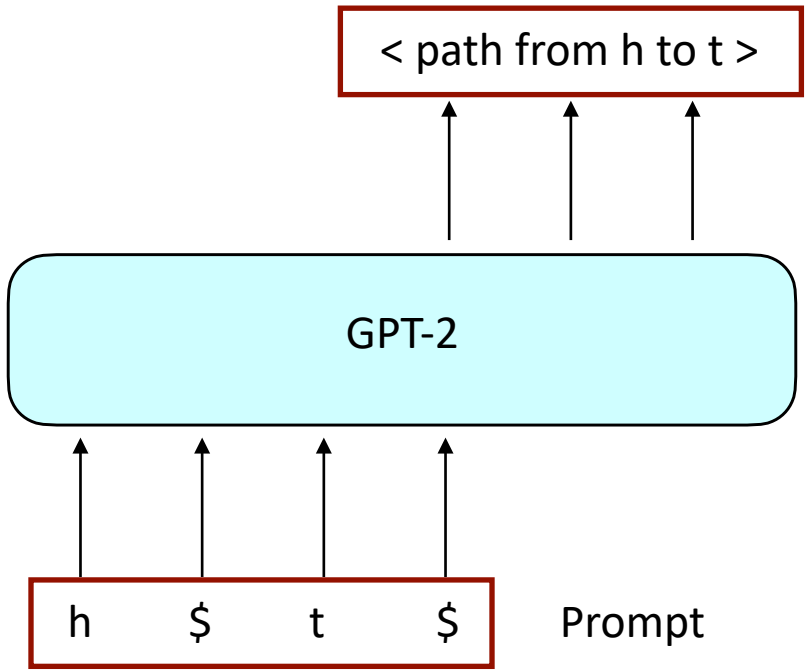
How to obtain edge features?

- **Solution**: Use GPT-2 to generate facts
- Use average of hidden states for generated fact as edge features
- Training: Fine-tune GPT-2 on synthetic facts from KG
- Inference: Use prompt [h, $, t, $] to generate the fact (**HGN w/ RelGen edges**)



How to obtain edge features?

- **Solution**: Use GPT-2 to generate **paths** from h to t [Wang et al. 2020]
- Use pooling over the path to obtain edge features (**HGN w/ PathGen edges**)



Model

- Graph Neural Networks (GNNs) for reasoning
- Update node representations using message passing.
- Node is updated according to information aggregated from the node's graph neighborhood.

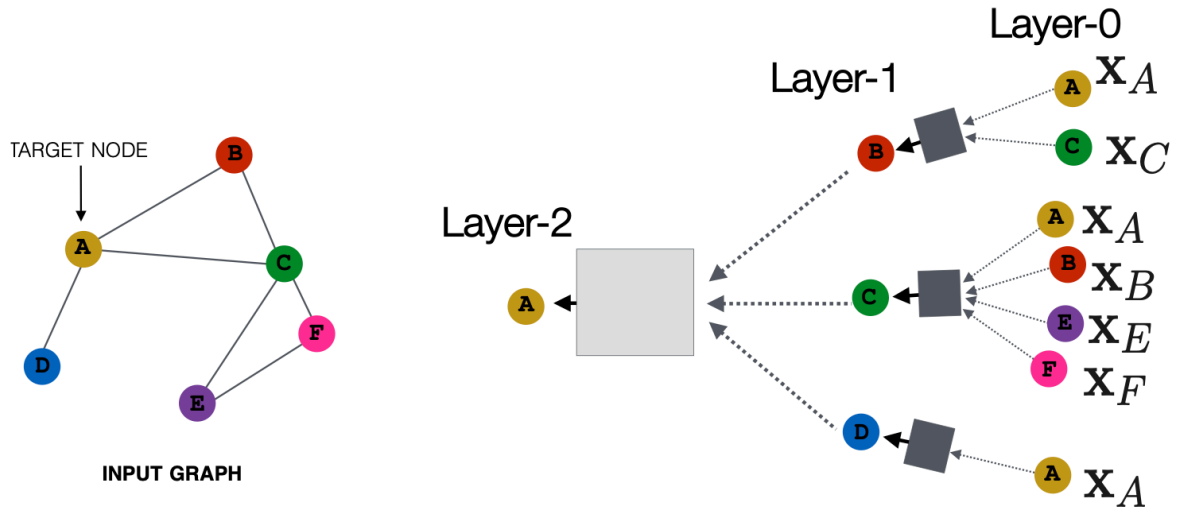


Figure credit: William Hamilton

Edge-to-node Message Passing

- Edge's weight scales information flow through that edge
- Edge weight -> Edge relevance

$$e \to v : \mathbf{u}_{(i,j)}^{\ell} = f_u^{\ell}\left(\left[\mathbf{h}_i^{\ell-1}, \mathbf{h}_{(i,j)}^{\ell}\right]\right);$$

$$\mathbf{h}_j^{\ell} = f_{e\to v}^{\ell}\left(\sum\nolimits_{i\in\mathcal{N}_j} \mathbf{A}_{(i,j)}^{\ell}\mathbf{u}_{(i,j)}^{\ell}\right)$$

- Edge weight $\mathbf{A}_{(i,j)}^{\ell}$

Node-to-edge Message Passing

- Embedding of an edge is influenced by
  - Neighboring nodes
  - Context s
- **Edge score** $w^{\ell}_{(i,j)}$
  - Measures edge's relevance to s
- Normalized edge scores -> Adjacency matrix

$$v \to e : \mathbf{h}^{\ell}_{(i,j)} = f^{\ell}_{v \to e}\left(\left[\mathbf{h}^{\ell-1}_{i}, \mathbf{h}^{\ell-1}_{j}, \mathbf{h}^{\ell-1}_{(i,j)}, \mathbf{s}\right]\right)$$

$$w^{\ell}_{(i,j)} = f^{\ell}_{w}\left(\left[\mathbf{h}^{\ell-1}_{(i,j)}, \mathbf{s}\right]\right)$$

$$\mathbf{A}^{\ell}_{(i,j)} = \frac{e^{w^{\ell}_{(i,j)}}}{\sum_{(s,t)\in\mathcal{E}'} e^{w^{\ell}_{(s,t)}}}$$

Model

- Iteratively update the node and edge representations using message passing

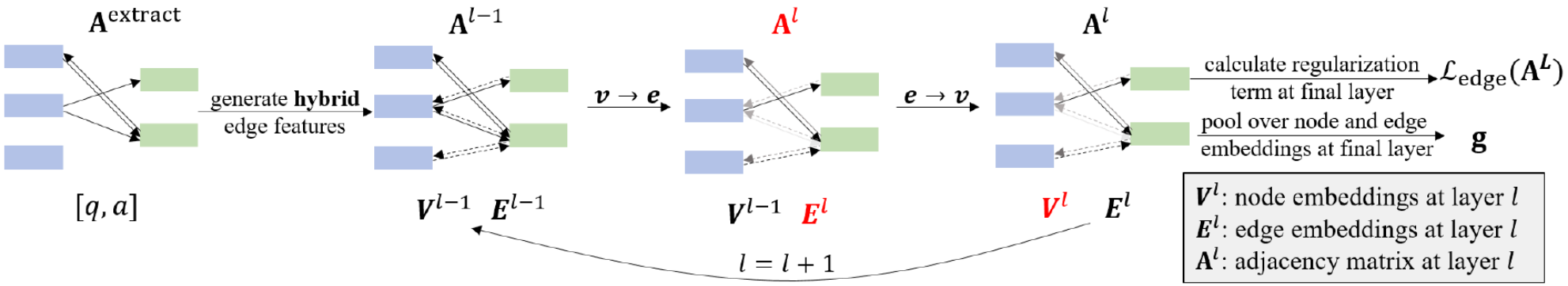


Figure 3: **Overview of HGN.** After building a hybrid graph of extracted and generated edges (§3.2), HGN reasons over the hybrid graph by updating the node embeddings $\mathbf{V}$, hybrid edge embeddings $\mathbf{E}$, and adjacency matrix $\mathbf{A}$ at each layer ℓ (§3.3). **Darker** edges indicate higher weights. **Red** variables are updated in the previous step.

Training

- Aggregate node embeddings via attentive pooling -> $\mathbf{v}_{\text{agg}}$
- Aggregate edge embeddings via edge-weighted sum pooling -> $\mathbf{e}_{\text{agg}}$
- **s** - LM encoding of question and answer.

$$\mathbf{g} = [\mathbf{v}_{\text{agg}}, \mathbf{e}_{\text{agg}}]$$

$$\rho(q, a) = f_{\text{score}}([\mathbf{s}, \mathbf{g}]; \boldsymbol{\theta}_{\text{score}})$$

$$\hat{\rho}(q, a) \propto \exp(\rho(q, a))$$

$$\mathcal{L}_{\text{task}}\left(\hat{\rho}(q, a; \boldsymbol{\theta})\right), y) = -y \log \hat{\rho}(q, a; \boldsymbol{\theta})$$

Entropy Regularization

- Penalize non-discriminative edge weights
- Low entropy -> Higher certainty about edge weights
- Prevents uniform distribution over edge weights

$$\mathcal{L}_{\text{edge}}(\mathbf{A}^L(q,a)) = -\sum_{(i,j):(v_i,v_j)\in\mathcal{E}'} \mathbf{A}^L_{(i,j)} \log \mathbf{A}^L_{(i,j)}$$

$$\mathcal{L}(\theta) = \sum_{(q,a,y)\sim X_{\text{train}}} \left[\mathcal{L}_{\text{task}}\left(\hat{\rho}(q,a)\right), y) + \beta \cdot \mathcal{L}_{\text{edge}}(\mathbf{A}^L(q,a))\right]$$

Commonsense QA results

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 (Velickovic et al., 2018)	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 (w/ PathGen edges)	55.68 (±0.29)	57.77 (±0.39)	58.19 (±0.27)	60.89 (±0.19)	70.95 (±0.21)	73.41 (±0.31)
HGN (w/ RelGen edges)	**55.72** (±0.32)	**58.01** (±0.29)	**58.19** (±0.11)	**61.11** (±0.21)	**71.10** (±0.11)	**73.64** (±0.30)

Results on Commonsense QA on in-house test set

Commonsense QA results

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 (w/ RelGen edges)	**77.3**	**80.0**
TeGBERT	76.8	-
ALBERT+PathGenerator (Wang et al., 2020)	75.6	78.2
ALBERT (Lan et al., 2020)	-	76.5

Table 2: **Leaderboard of CommonsenseQA.** HGN ranks first among comparable systems, especially achieving remarkable improvement over PathGenerator (Wang et al., 2020).

CODAH results

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 (Velickovic et al., 2018)	65.88	82.78
GN (Battaglia et al., 2018)	65.52	82.06
GconAttn (Wang et al., 2019a)	65.17	82.35
MHGRN (Feng et al., 2020)	65.92	83.07
PathGenerator (Wang et al., 2020)	64.67	82.27
HGN (w/ PathGen edges)	66.21	**84.32**
HGN (w/ RelGen edges)	**66.75**	84.08

Table 3: **Test accuracy on CODAH.** Both our model variants consistently improve over all baselines. We use the official split for 5-fold cross validation. Mean accuracy on 5 folds are presented.

Openbook QA and QASC results

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)	78.05 (±0.77)	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 (Velickovic et al., 2018)	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 (w/ PathGen edges)	80.05 (±0.54)	74.10 (±0.42)
HGN (w/ RelGen edges)	**80.15** (±0.38)	**74.27** (±0.31)

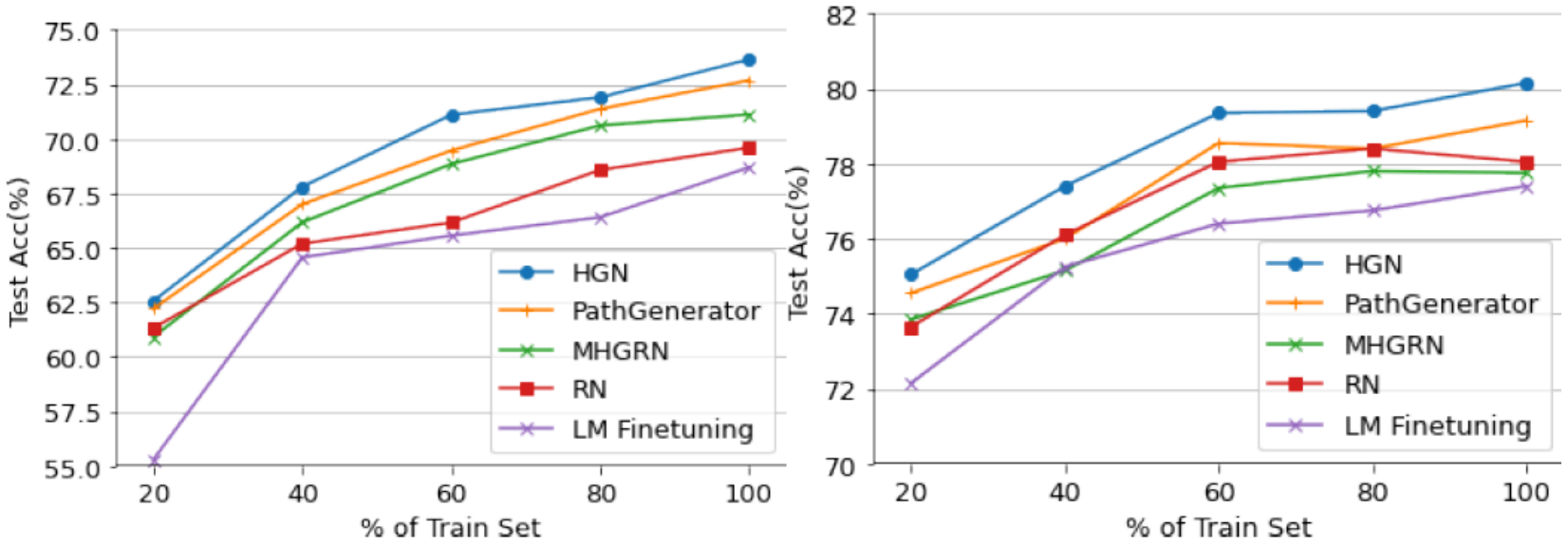
Table 4: **Test accuracy on OpenBookQA and QASC with retrieval-augmented methods as base models.** Both our model variants greatly improve over all baselines except HGN (w/ PathGen edges) over MHGRN. For OpenbookQA baselines with *, we use reported numbers from Wang et al. (2020). Mean and standard deviation of four seeds are presented.

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 (w/ RelGen edges)	**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 5: **Leaderboard of OpenBookQA.** Our HGN ranks first among all submissions using AristoRoBERTa as the text encoder.

Low-resource setting

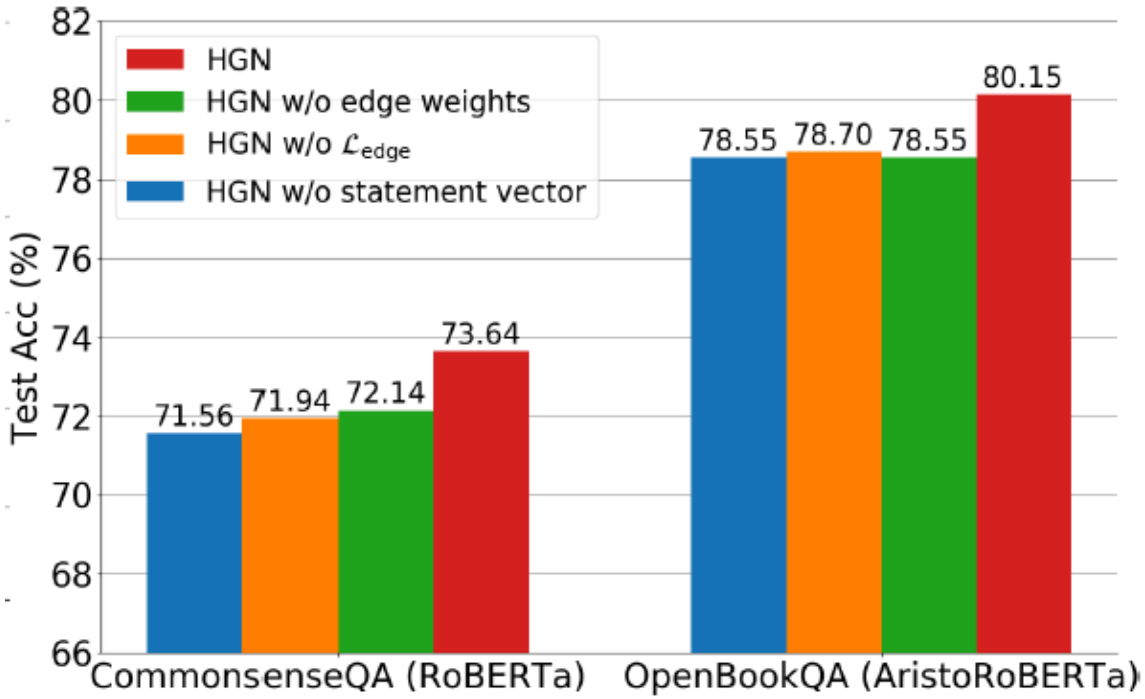
- Good performance in low-data regime
- Performance gap with LM fine-tuning is more in case of Commonsense QA
  - Retrieval makes up for low data in case of OpenBook QA



(a) CommonsenseQA (RoBERTa). (b) OpenBookQA (AristoRoBERTa).

Ablation Studies

- Model variants
  - HGN
  - HGN w/o edge weights
  - HGN w/o entropy regularization
  - HGN w/o statement vector



(c) Ablations on model variants.

User study on Learned Graph Structures

- Binarize the learned adjacency matrix
- Evaluation metrics
  - Validness (0 or 1)
  - Helpfulness (0-2)
- Prune rate

$$1 - \frac{\#\text{ edges in binarized } \mathbf{A}^L}{\#\text{ edges in } \mathbf{A}^0}$$

Contextualized Graph	**GN** ($\mathbf{A}^{\text{extract}}$)	**HGN** ($\mathbf{A}^L$)
Number of Edges	3.65 ($\pm$2.73)	4.38 ($\pm$3.24)
Number of Valid Edges	2.67 ($\pm$1.95)	3.15 ($\pm$1.98)
Percentage of Valid Edges	71.64%	78.51%
Average Helpfulness Score of Edges	0.90 ($\pm$0.50)	1.16 ($\pm$0.51)
Prune Rate	-	22.84%

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

Case studies

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

Graph of HGN

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

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

Graph of GN

(a) **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

Graph of GN

(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

Graph of HGN

(b) **Case II: Helpful generated facts are incorporated.**

Thank You!