Graph Neural Prompting with Large Language Models

AAAI 2024

Table of Contents

Background & Motivation

- **Limitations of LLMs:** struggle to faithfully reproduce factual knowledge.
- **Prior attempts**
  - Joint KG + LLM pre-training → prohibitively expensive.
  - Directly injecting KG triples → noisy context degrades accuracy.
- **Industrial need:** Leverage **already-trained LLMs** without full fine-tuning.

Research Question & Novelty

> **Can we learn useful knowledge from a KG and inject it into a frozen LLM with minimal extra parameters?**

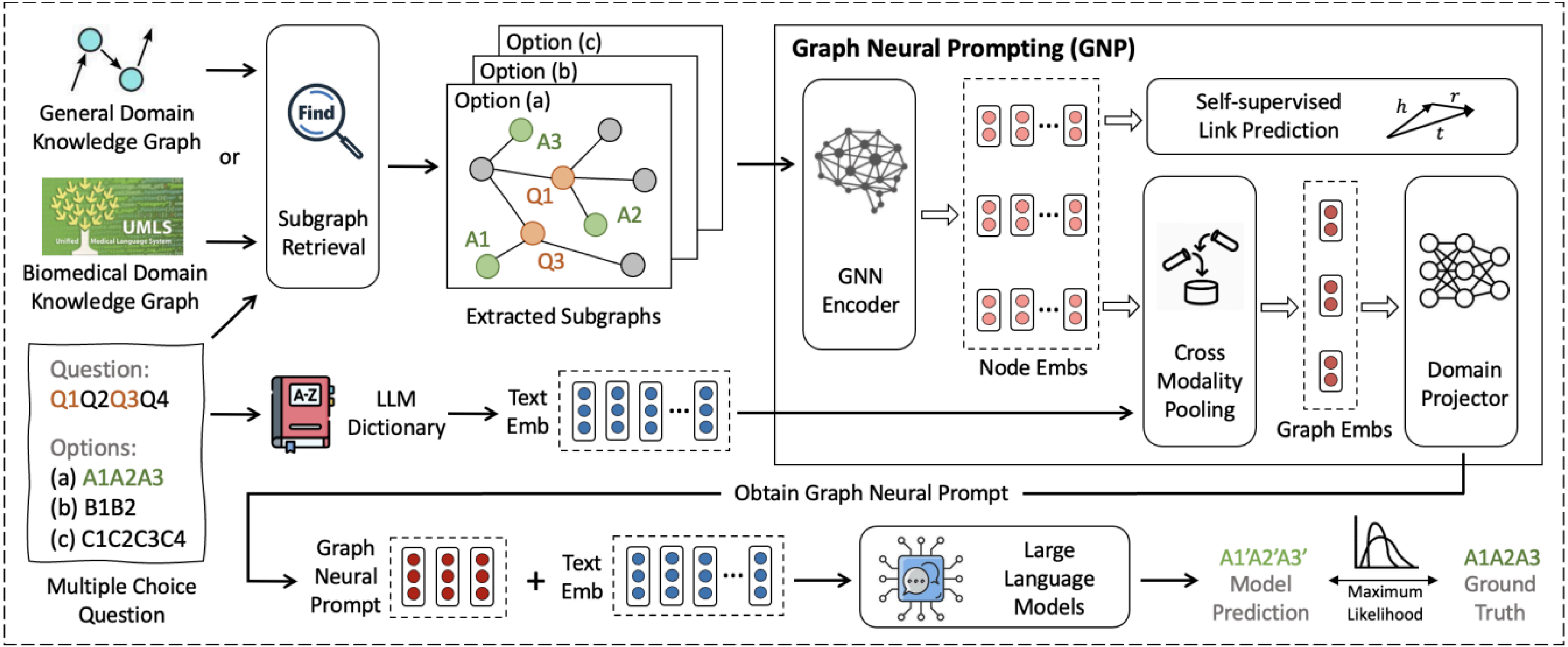
Key Contributions

- Introduce **Graph Neural Prompting (GNP)** – a lightweight, *plug-and-play* adapter.
- Combine GNN encoding, cross-modal pooling, a domain projector, and self-supervised link prediction.
- Achieve **+13.5 %** average accuracy on commonsense & biomedical QA with *no* LLM updates; still adds **+1.8 %** when paired with LoRA.

Proposed Method: GNP

- **Step 1**: Retrieve 2-hop subgraph around question/answer entities.
- **Step 2**: Encode subgraph with a Graph Attention Network.
- **Step 3**: *Cross-modal pooling* selects nodes most relevant to the text.
- **Step 4**: *Domain projector* maps graph embedding into LLM space → **Graph Neural Prompt**.
- **Step 5**: Prepend prompt to LLM input; train with QA loss + link-prediction loss.

Proposed Method : GNP



Step 1 – Subgraph Retrieval

1. Identify key entities • Scan the question and every answer choice for surface strings that match KG labels (simple fuzzy match).
2. Grow a mini-graph (≤ 2 hops) • Collect neighbours up to two steps away for each matched entity. • Skip ultra-generic hubs like thing or entity to avoid noise.
3. Trim for efficiency • Stop expansion once the subgraph hits roughly 200 nodes / 600 edges. • Result: a compact, question-focused "pocket KG" that fits easily in GPU memory.

Step 2 – GNN Encoder

- Backbone: **Graph Attention Network (GAT)**, *2 ≤ L ≤ 3* layers.
- Each layer computes:
  $$h_i^{(l+1)} = \mathrm{ReLU}\big(\mathrm{AttnAgg}(h_{\mathcal{N}(i)}^{(l)})\big)$$
- Output: contextual node embeddings H_1 © KG structure.

Step 3 – Cross-Modality Pooling

1. **Self-Attention** ranks nodes inside the subgraph.
2. **Cross-Attention** with LLM token embeddings *T*:
   $A = \mathrm{softmax}\big(H_2\, T'^{\top} / \sqrt{d_g}\big)$
3. Produce question-aware node set H_3 → mean-pool to vector H_4.

- Effect: suppress irrelevant KG parts, highlight useful clues.

Step 4 – Domain Projector

- 2-layer FFN aligns spaces: $d\_g \rightarrow d\_t$ (LLM hidden size).
- Adds non-linearity + LayerNorm.
- Resulting **Graph Neural Prompt Z** has same dimensionality as a token embedding → can be *prepended*.

Step 5 – Self-Supervised Link Prediction

- **Edge masking** – randomly hide 20 % of triples inside each instance-specific subgraph.
- Mask subset of edges; predict with **DistMult** score ϕ(h,r,t).
- **Two losses**
  • $\mathbf{L_{QA}}$ – cross-entropy that forces the frozen LLM (+ prompt) to pick the right answer choice.
  • $\mathbf{L_{link}}$ – margin-ranking loss that drives true triples above negatives, preserving KG structure.
- **Joint objective**:
  $L = L_{\mathrm{QA}} + \lambda L_{\mathrm{link}}, \quad \lambda = 0.1$ (commonsense) / 0.5 (biomedical).
- **Why it helps** – the auxiliary link term maintains discriminative node embeddings and counteracts any over-smooth bias introduced by shallow GAT + pooling.

Component Breakdown

Component	Purpose
GNN Encoder	Capture KG structure & semantics
Cross-modal Pooling	Filter noise, highlight text-relevant nodes
Domain Projector	Align graph and text embedding spaces
Self-supervised Link Prediction	Preserve structural knowledge during training

Experimental Setup

- **Datasets (QA)**
  - *Commonsense*: OBQA, ARC, PIQA, RiddleSense (KG: ConceptNet)
  - *Biomedical*: PubMedQA, BioASQ (KG: UMLS)
- **Models**
  - FLAN-T5-3B & 11B (frozen)
  - Prompt Tuning, LoRA, Full FT baselines
- **Training**
  - Batch size 32, learning rate 5e-4, λ=0.3 for link-prediction loss

Main Results (Avg. over 6 tasks)

Setting	Accuracy	Δ vs. Baseline
Prompt Tuning	65.9 %	–
GNP (frozen)	**74.4 %**	**+12.7 pp**
LoRA	76.2 %	–
LoRA + GNP	**77.4 %**	**+1.6 pp**
Full FT	76.8 %	–0.6 pp

Ablation Study

- **Remove Cross-modal Pooling** → −4.8 pp
- **Remove Domain Projector** → −7.1 pp *(most critical)*
- **Remove Link Prediction** → −3.2 pp

Case Study: OBQA Example

> *Question*: What keeps Earth in orbit around the Sun?
>
> *GNP-selected subgraph* focuses on nodes "gravity", "Sun", "Earth", filtering unrelated edges → LLM correctly answers *gravity*.

Discussion

- GNP consistently benefits both **frozen** and **lightly-tuned** LLMs.
- Cost-effective: <1% of LLM parameters, no extra inference latency.
- Orthogonal to other adapter methods – can be stacked.

Conclusion & Future Work

- **GNP** offers a practical path to fuse structured knowledge into off-the-shelf LLMs.
- Outperforms full fine-tuning in 10/12 evaluations while using *far* less compute.
- **Next steps**
    - Extend to multilingual KGs & tasks.
    - Explore dynamic graph retrieval at inference time.
    - Combine with retrieval-augmented generation for open-domain QA.