# Graph Machine Learning

## Lecture 10 : GNNs Meet Large Language Models

Oct 26 2026


Xavier Bresson

https://x.com/xbresson

Department of Computer Science

National University of Singapore (NUS)

# Course lectures

- Introduction to Graph Machine Learning
- Part 1: GML without feature learning (before 2014)
  - Introduction to Graph Science
  - Graph Analysis Techniques without Feature Learning
    - Graph clustering
    - Graph SVM
    - Recommendation on graphs
    - Graph-based visualization
- Part 2 : GML with shallow feature learning (2014-2016)
  - Shallow graph feature learning
- Part 3 : GML with deep feature learning, a.k.a. GNNs (after 2016)
  - Graph Convolutional Networks (spectral and spatial)
  - Weisfeiler-Lehman GNNs
  - Graph Transformer
  - → GNNs Meet Large Language Models
  - Graph generation & molecular science
  - Material design
  - Deep Learning for Combinatorial optimization

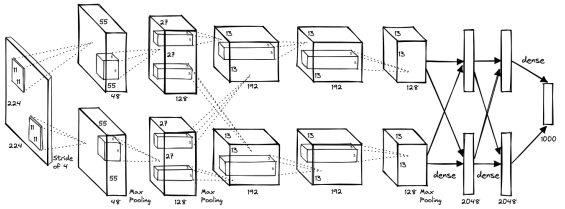
# Outline

- LLMs and GenAI
- Do LLMs make GNNs obsolete ?
- Combining LLMs and GNNs
- Using LLM for GNN reasoning
- Using GNN for LLM reasoning
- Conclusion

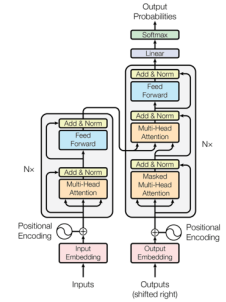
# Outline

- **LLMs and GenAI**
- Do LLMs make GNNs obsolete ?
- Combining LLMs and GNNs
- Using LLM for GNN reasoning
- Using GNN for LLM reasoning
- Conclusion

# Two waves of the deep learning revolution



- Computer vision revolution
  - Year: 2012, Dataset: ImageNet (1.2M 224x224 labeled images)
  - Architecture: CNNs/AlexNet[1]: 8 layers, 61M parameters
  - Hardware: 2 x NVIDIA GTX 580 GPUs (2 x 1.6 teraflops for single-precision)
  - Training time: 5 days
  - Industry: Search, ad recommender (Facebook, TikTok), tagging, autonomous driving (Waymo), surveillance (real-time face monitoring).
- Language model revolution
  - Year: 2022, Dataset: Internet and third-party (300B word tokens)
  - Architecture: Decoder-only Transformer (GPT-3.5[2], 96 layers, 175B parameters)
  - Hardware: 10,000 x NVIDIA V100 GPUs ($10^5$ training teraflops)
  - Training time: 15 days
  - Industry: GenAI for everyday tasks s.a. coding, summarization, content creation, dialogue systems (e.g. ChatGPT, Claude, Gemini).
  - Paradigm: Large pre-trained networks on large-scale datasets are called foundation models.




[1] Krizhevsky-Sutskever-Hinton, ImageNet Classification with Deep Convolutional Neural Networks, 2012
[2] Brown et-al, Language Models are Few-Shot Learners, 2020

# GenAI industry (Oct 2026)

- Text generation[1] :
  - Proprietary models : GPT-6 Astra (Sep 2026), Gemini 4 Argon (Sep 2026), Claude Fable 5.1 (Sep 2026), Grok 4.7 (Sep 2026), Muse Wonder Full (Oct 2026).
  - Open-weight models : Meta Muse Spark Open (Oct 2026), Google Gemma 4 31B (Mar 2026), DeepSeek-V4-Pro (Apr 2026), Qwen3.8-2.4T-A95B (Aug 2026), Llama 4 Maverick (Apr 2025).
  - Data : Llama 4 Maverick was pre-trained on 22T multimodal tokens (text, images, video), DeepSeek-V4-Pro on 32T tokens. But training data sizes for other models remain undisclosed.
  - Architecture : Transformers, using dense or Mixture-of-Experts (MoE), sometimes with multimodal early fusion. Total/active parameters: Muse Spark Open 816B/42B, DeepSeek-V4-Pro 1.6T/49B, Qwen3.8 2.4T/95B, Llama 4 Maverick 400B/17B, Gemma 4 31B.
  - Compute: Frontier AI infrastructure spans O(100,000) NVIDIA GPUs with gigawatt-scale clusters. Exact model-level training compute is often undisclosed.
    xAI's Colossus reached 200,000 × H100s (Feb 2025).

| | |
|---|---|
| SYSTEM PROMPT (HUMAN-WRITTEN) | *In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.* |
| MODEL COMPLETION (MACHINE-WRITTEN) | The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.<br>Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.<br>Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow. |

[1] https://arena.ai/leaderboard

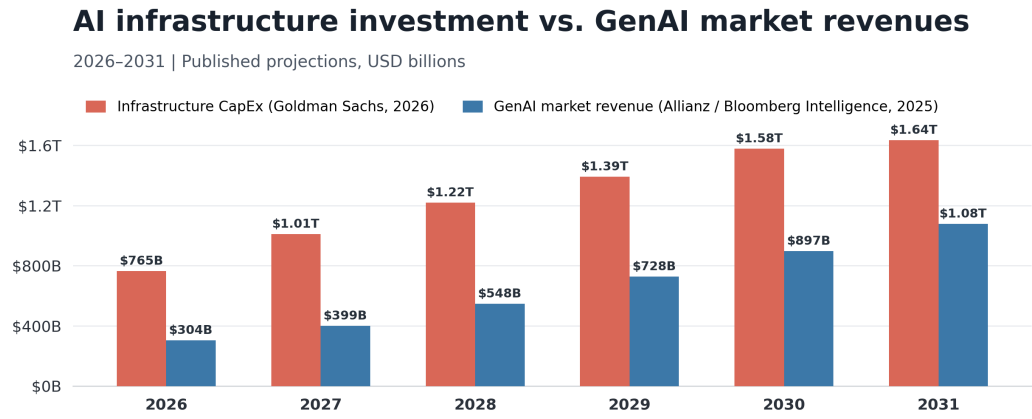
# GenAI industry (Oct 2026)

- Text-to-image generation[1] :
  - Proprietary/API models : Midjourney V8.2 (Jul 2026), GPT-Image-2.5 Flare/Sunburst (Sep 2026), Google Nano Banana 2 (Feb 2026), FLUX.2 pro (Nov 2025)/ max (Dec 2025).
  - Open-weight models : FLUX.2 dev (Nov 2025) / klein (Jan 2026), Ideogram 4.0 (Jun 2026), Qwen-Image-2.1 (Sep 2026), Stable Diffusion 3.5 (Oct 2024).
  - Data : Frontier training datasets remain undisclosed. LAION-5B (Mar 2022), containing 5.85B image-text pairs, is a historical public reference but likely not representative of current frontier training datasets.
  - Architecture : Diffusion/flow-matching Transformers. Ideogram 4.0 (Jun 2026) uses a 9.3B-parameter, 34-layer Diffusion Transformer (DiT) with structured captions.
  - Compute : Current frontier image-model training compute is undisclosed. For historical context, Stability AI operated a cluster of 4,000 × NVIDIA A100 GPUs (2022).

[1] https://artificialanalysis.ai/image/leaderboard/text-to-image

# GenAI revenues vs. investment (Oct 2026)

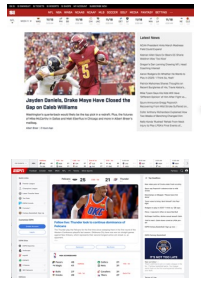
- OpenAI[1] : Reuters reported US$50B in 2025 revenue. Anthropic[2] : Q3 2026 revenue was $16.8B. Google[3] : Alphabet's Q3 2026 report confirms $24.8B in Google Cloud revenue.
- Most profitable GenAI application categories (Oct 2026)[4] :
  - General-purpose AI assistants : $7.2B (enterprise spending) with ChatGPT, Claude, Microsoft Copilot.
  - Coding : $4.0B for GitHub Copilot, Cursor, Claude Code, Codex.
  - Healthcare : $1.5B for AI medical scribes, clinical documentation.
  - AI agents : $750M for business workflow and task automation.
- Goldman Sachs[5] estimates companies, governments and organizations worldwide will spend $0.8T on AI-related infrastructure in 2026, and will generate $0.3T revenue.
- In 2031 (5 years), the projected AI infrastructure will reach $1.6T, and $1.1T revenue.



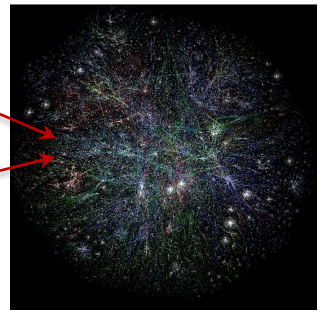

[1] https://www.reuters.com/business/openais-annualized-revenue-20-billion-less-than-previously-signaled-ft-reports-2026-10-08
[2] https://www.reuters.com/business/finance/anthropics-q3-revenue-surges-45-nearly-17-billion-2026-10-14
[3] https://www.sec.gov/Archives/edgar/data/1652044/000165204426000066/googexhibit991q22026.htm
[4] https://menlovc.com/perspective/2026-the-state-of-consumer-ai
[5] https://www.goldmansachs.com/insights/articles/tracking-trillions-the-assumptions-shaping-scale-of-the-ai-build-out

# LLMs are trained from graph-structured data

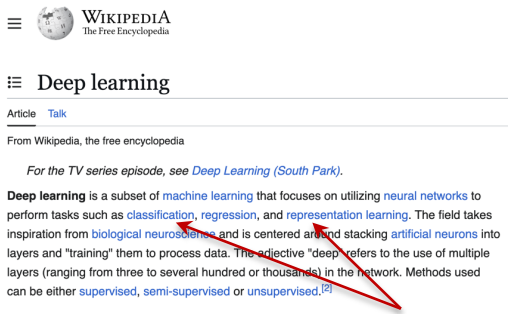
- The Web can be viewed as a graph : Webpages are nodes and hyperlinks define relationships between them, producing an enormous, heterogeneous network of information.
- Wikipedia forms a large hyperlink network, while Wikidata represents entities and their relations explicitly as a structured knowledge graph.
- LLMs are trained mainly on text extracted from these sources, rather than on the graph itself.
- Nevertheless, textual data encodes many relations, and LLMs can reason over graphs when their nodes and edges are represented in text -- for example, classification or link prediction.
- This raises the critical question : Can LLMs reason reliably over graph-structured data, or do we still need GNNs to explicitly exploit the underlying graph structure?



Text-based nodes

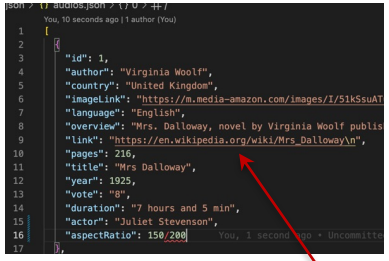


Web




Wikipedia

Hyperlinks




Json file

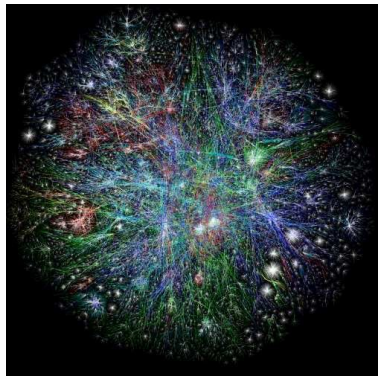
Links to other content

# Outline

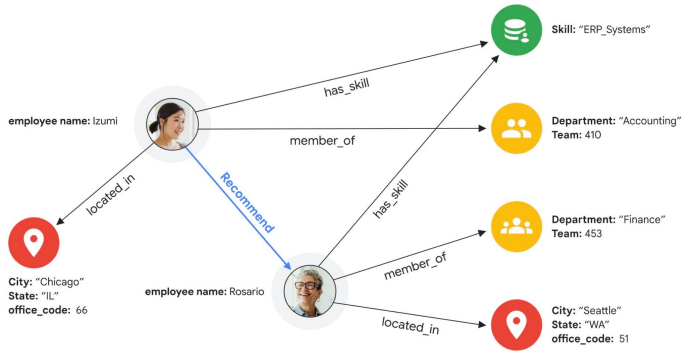
- LLMs and GenAI
- **Do LLMs make GNNs obsolete ?**
- Combining LLMs and GNNs
- Using LLM for GNN reasoning
- Using GNN for LLM reasoning
- Conclusion

# Text-attributed graphs

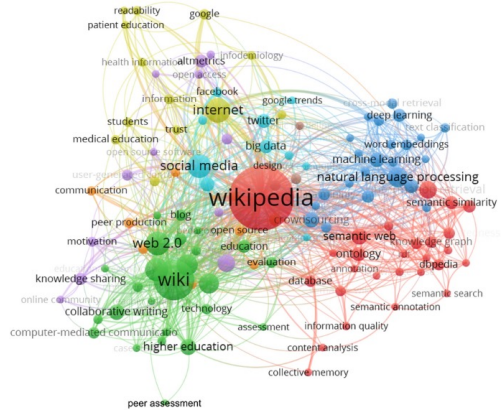
- We consider text-attributed graphs (TAGs), where nodes are associated with natural language.
- We exclude graphs without textual attributes, such as molecular graphs.



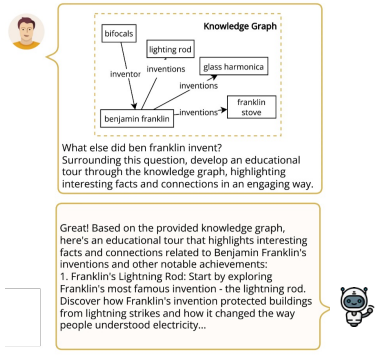
Web



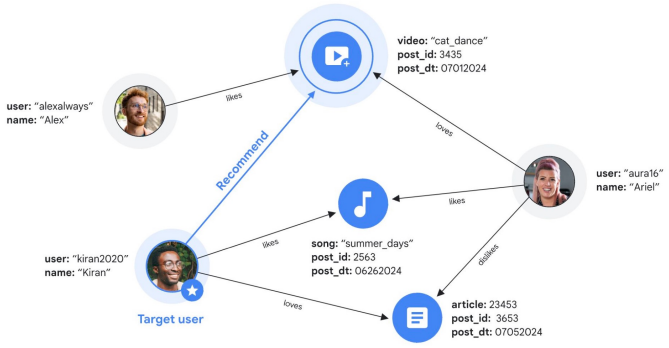

Knowledge graph



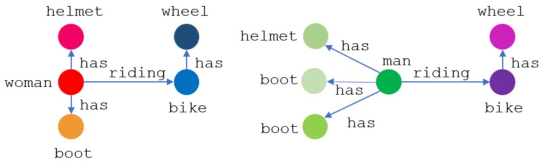

Wikipedia




QA graph



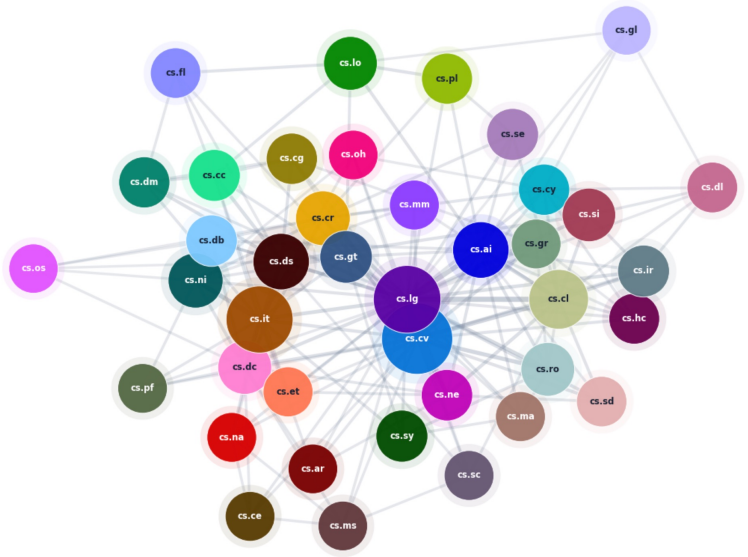

Social graph

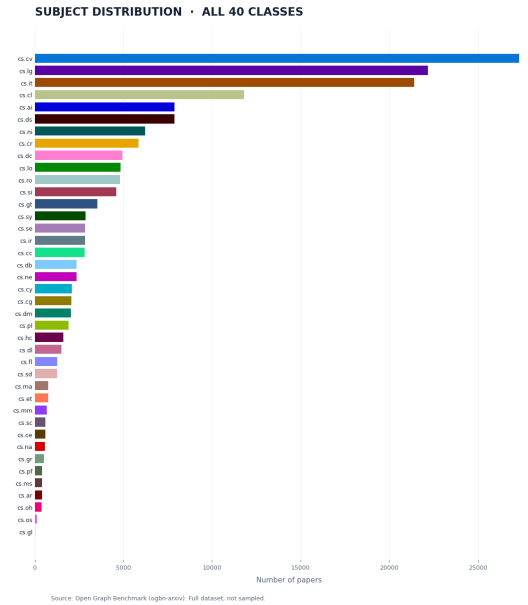



Scene graph

# A canonical TAG benchmark : ogbn-arxiv

- Consider ogbn-arxiv[1], a widely used node-classification benchmark from OGB.
- Graph : 169,343 papers, 1,166,243 directed citation edges, 40 classes.
- Text : Each paper/node has a title and abstract.
- Task : Predict each paper's primary arXiv CS category s.a. cs.AI, cs.LG, cs.OS.
  The train/validation/test split is chronological : $\leq$ 2017 / 2018 / $\geq$ 2019.




[1] Hu-etal, Open graph benchmark: Datasets for machine learning on graphs, 2020

# Do LLMs make GNNs obsolete ?

- Results
  - RevGAT[1] + OGB shallow text features achieves 70.8% test accuracy.
  - The 2022-SOTA GLEM[2] (LM+GNN) achieves 76.6% test accuracy.
  - Zero-shot GPT-3.5[3], using only title + abstract, achieves 73.5% accuracy.
  - Conclusion : GNN alone: 70.8 $<$ LLM alone: 73.5 $<$ LM+GNN: 76.6
  - Thus, the LLM outperforms the shallow-feature GNN, but combining language and graph structure performs best!
  - Additionally, ogbn-arxiv papers may appear in GPT-3.5's training data, as the dataset predates GPT-3.5's training cutoff (Sep 2021). Therefore, potential data leakage is possible.

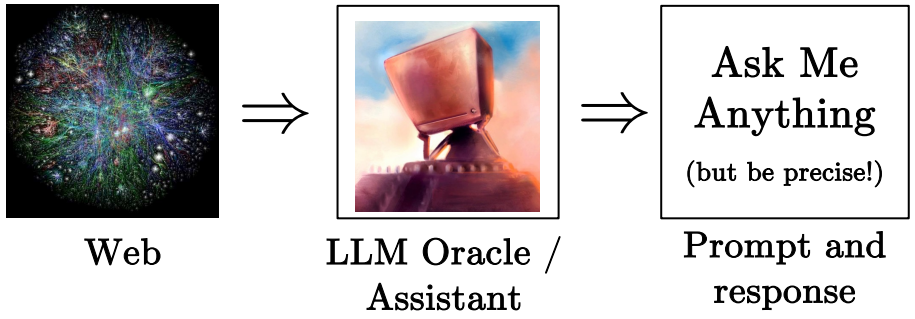
[1] Li-etal, Training graph neural networks with 1000 layers, 2021
[2] Zhao-etal, Learning on large-scale text-attributed graphs via variational inference, 2022
[3] Brown-etal, Language Models are Few-Shot Learners, 2020

# Outline

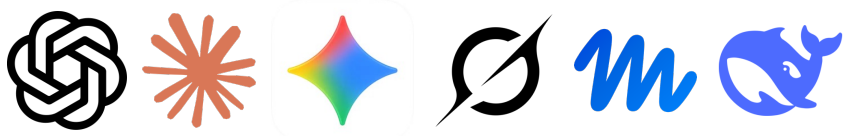
- LLMs and GenAI
- Do LLMs make GNNs obsolete ?
- **Combining LLMs and GNNs**
- Using LLM for GNN reasoning
- Using GNN for LLM reasoning
- Conclusion

# LLMs : strengths and limitations

- Strengths
  - Learn rich distributions over token sequences, enabling fluent generation and contextual prediction.
  - Pretrained on massive, heterogeneous corpora.
  - Strong zero-/few-shot adaptation through prompting and in-context learning[1].
  - Scaling continues to improve performance[2] (Oct 2026) : More training compute improves models, while post-training and test-time compute provide additional scaling axes.

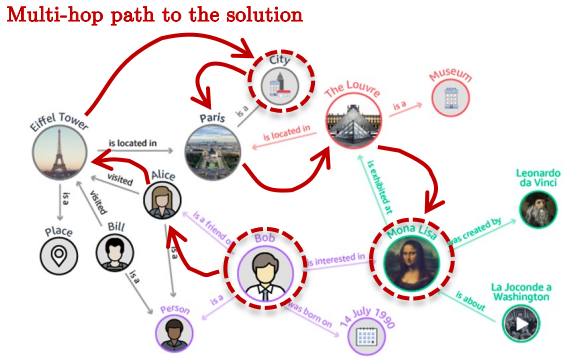



- Limitations
  - Can hallucinate, producing plausible but incorrect or unsupported claims.
  - Reasoning has improved greatly with modern reasoning models, but remains compute-intensive on difficult tasks.
  - Graph reasoning remains challenging as graph topology encoded as text is not natural for LLMs.



[1] Brown et-al, Language models are few-shot learners, 2020
[2] Sutton, the bitter lesson, 2019

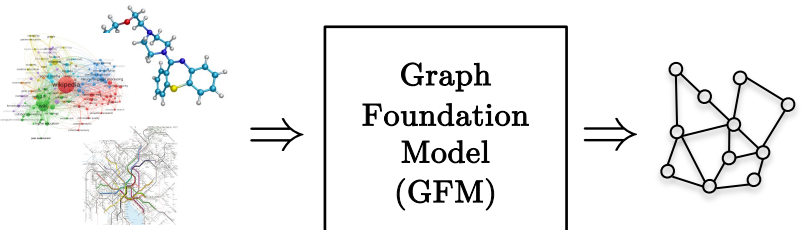
# GNNs : structure-aware, but hard to scale




Q: Is the Mona Lisa in the same city as Alice's friend Bob?

- Strengths
  - Designed for relational reasoning by aggregating information over multi-hop neighborhoods.
  - Effective on graph-structured data across domains: recommendation, fraud, molecules, knowledge graphs, physical systems, etc.
  - AlphaFold2, recognized by the 2024 Chemistry Nobel, uses pairwise and triangle-based reasoning to predict accurate 3D protein structures.

- Limitations
  - No graph foundation model [1,2,3] yet matches LLMs/VLMs in scale, breadth and transferability.
  - Graph training data is fragmented across domains and lacks a Web-scale universal corpus.
  - Sparse, irregular graph operations are less accelerator-friendly than dense matrix operations.
  - Pretrained GNNs are generally much smaller (< 50M param) than frontier LLMs / VLMs.
  - GNNs see substantial domain-specific use, but foundation GNN+LLM systems remain relatively uncommon as unclear profitable products[4].

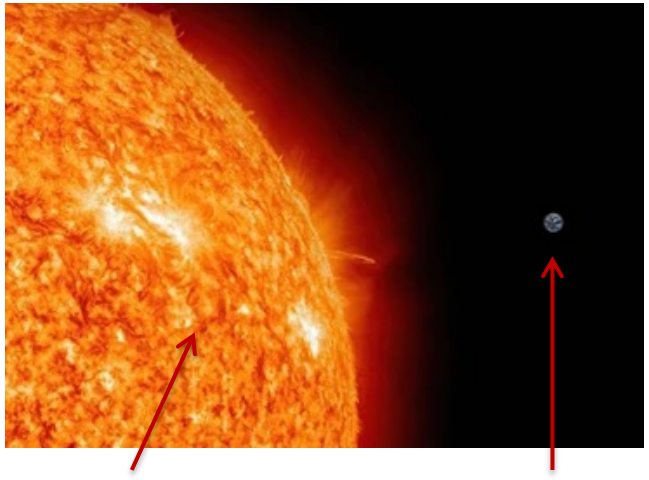




[1] Li et-al, A Survey of Graph Meets Large Language Model: Progress and Future Directions, 2024
[2] Liu et-al, One for All: Towards Training One Graph Model for All Classification Tasks, 2024
[3] Mao et-al, Graph Foundation Models are Already Here, 2024
[4] https://www.reddit.com/r/MachineLearning/comments/1c9vibh/d_why_isnt_gnn_in_high_demand_in_industry

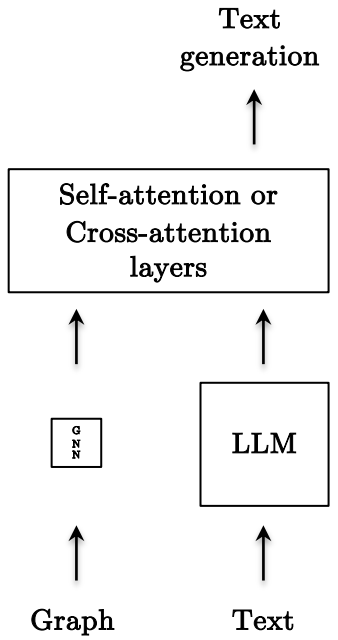
# The challenge of combining LLMs and GNNs

- Joint graph-text foundation models are a promising direction for combining semantic and structural reasoning.
- A key challenge is aligning two very different representations : Dense, sequential token representations and sparse, irregular graph structures. This challenge is further compounded by a large disparity in training scales, spanning approximately five orders of magnitude.
- Effective integration therefore requires scalable graph encoding, cross-modal alignment, and efficient mechanisms for exchanging information between LLMs and GNNs.



LLMs
$10^{11}$-$10^{12}$ param
$10^{13}$ training tokens

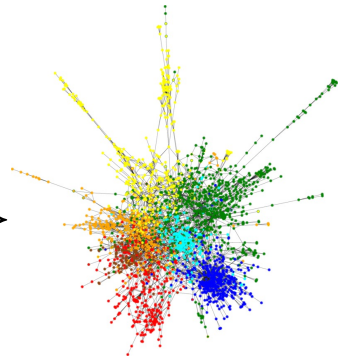
GNNs
$10^{7}$-$10^{8}$ parameters
$10^{8}$ nodes in OGB



# Task-specific LLM+GNN integration

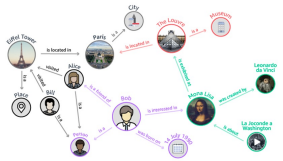
- The key question is : How should the two models interact for each task?
- We seek tasks where combining a LLM with a small GNN provides complementary benefits.
- Such task-specific combinations can :
  - Transfer semantic knowledge from LLMs to improve learning on small text-attributed graphs and low-data GNN tasks[1].
  - Use graph structure and retrieved facts to ground LLM generation in a more constrained and verifiable evidence space, hence reducing hallucinations[2].

Frontier LLM GPT-3.5
175G parameters 300B
word tokens



Cora graph GCN
2,708 nodes
23K parameters

TAPE



Small
Knowledge graph
100 nodes

Open-weight LLM Llama 2
7G parameters
2T word tokens

G-Retriever

# Outline

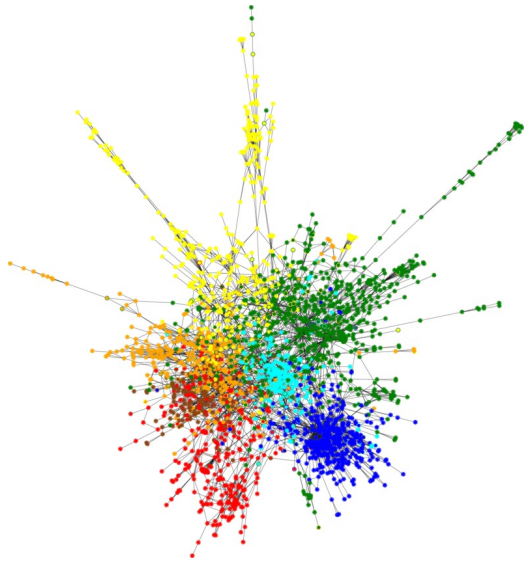
- LLMs and GenAI
- Do LLMs make GNNs obsolete ?
- Combining LLMs and GNNs
- **Using LLM for GNN reasoning**
- Using GNN for LLM reasoning
- Conclusion

# TAPE : LLM-enhanced node features for GNNs

- Goal : Improve GNNs using node features enriched with LLM-generated predictions and explanations[1].
- Key question : How can LLM outputs be transformed into informative features for downstream TAG tasks?

Frontier LLM
In 2023, we used GPT-3.5
175G parameters
300B word tokens

$\Rightarrow$

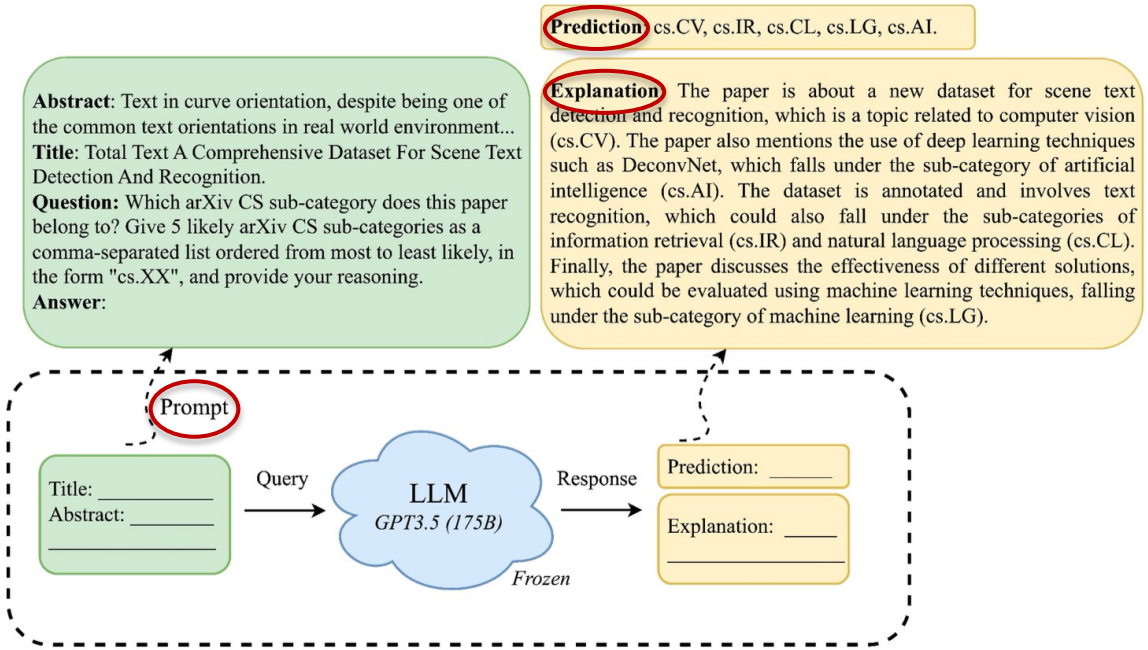


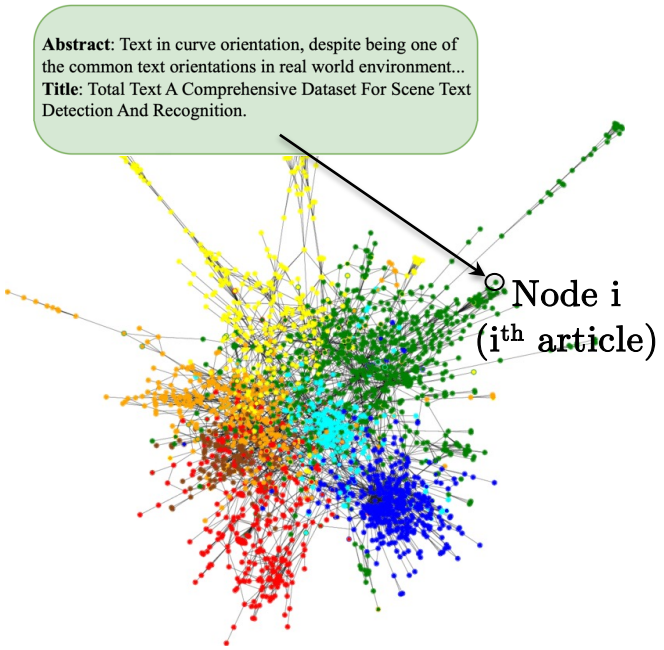
Cora graph GNN
2,708 nodes
7 classes
1.8M parameters

[1] He-LeCun-Bresson-etal, Harnessing Explanations: LLM-to-LM Interpreter for Enhanced Text-Attributed Graph Representation Learning, 2023

# LLM-generated predictions and explanations

- Extract LLM knowledge from the textual attributes of each TAG node.
- Given a node description (here title and abstract), prompt a frozen LLM to generate :
  - A zero-shot prediction for the downstream task.
  - A language explanation supporting that prediction.




Cora graph GNN
2,708 nodes
1.8M parameters

# LLM-to-LM interpreter : From text to node features

- Convert the prompt text (title, abstract) and LLM-generated explanation into fixed d-dimensional embeddings using a smaller, task-specific language model. Additionally, encode the ranked prediction separately.
- These complementary representations capture the original text and additional knowledge supplied by the LLM, providing enriched, frozen node features that will subsequently improve the downstream GNN's performance on TAG-related tasks.

**Abstract**: Text in curve orientation, despite being one of the common text orientations in real world environment... **Title**: Total Text A Comprehensive Dataset For Scene Text Detection And Recognition.

**Explanation**: The paper is about a new dataset for scene text detection and recognition, which is a topic related to computer vision (cs.CV). The paper also mentions the use of deep learning techniques such as DeconvNet, which falls under the sub-category of artificial intelligence (cs.AI). The dataset is annotated and involves text recognition, which could also fall under the sub-categories of information retrieval (cs.IR) and natural language processing (cs.CL). Finally, the paper discusses the effectiveness of different solutions, which could be evaluated using machine learning techniques, falling under the sub-category of machine learning (cs.LG).

**Prediction**: cs.CV, cs.IR, cs.CL, cs.LG, cs.AI.

Sequence-to-vector

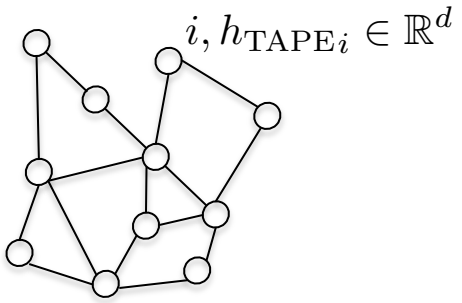
$\Rightarrow \quad h_{\text{orig}} \in \mathbb{R}^d$

$\Rightarrow \quad h_{\text{expl}} \in \mathbb{R}^d$

$\Rightarrow \quad h_{\text{pred}} \in \mathbb{R}^{n_c}$

These vectors will enhance the expressivity of node features

$\Rightarrow$

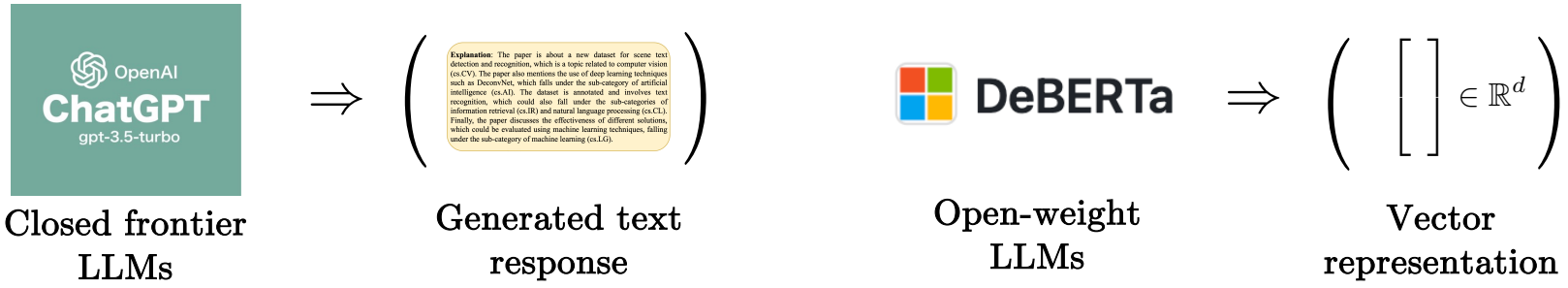



Text-attributed graph (TAG)

$h_{\text{TAPE}} = \{h_{\text{orig}}, h_{\text{expl}}, h_{\text{pred}}\}$

# From closed LLM outputs to vector representations

- We propose a sequence-to-vector approach combining a closed LLM with an open-weight LM.
- A closed-to-open LM interpreter converts generated text into a d-dimensional vector that summarizes its semantic information.
- Closed frontier LLMs such as GPT-3.5 (2022)[1] provide high-quality generated responses, while their internal hidden states are generally inaccessible through public APIs.
- Open-weight LM encoders s.a. DeBERTa[2] provides direct access to its internal hidden states.
- We therefore encode the closed-LLM response with an open-weight LM and use its pooled token representation as $h \in \mathbb{R}^d$.

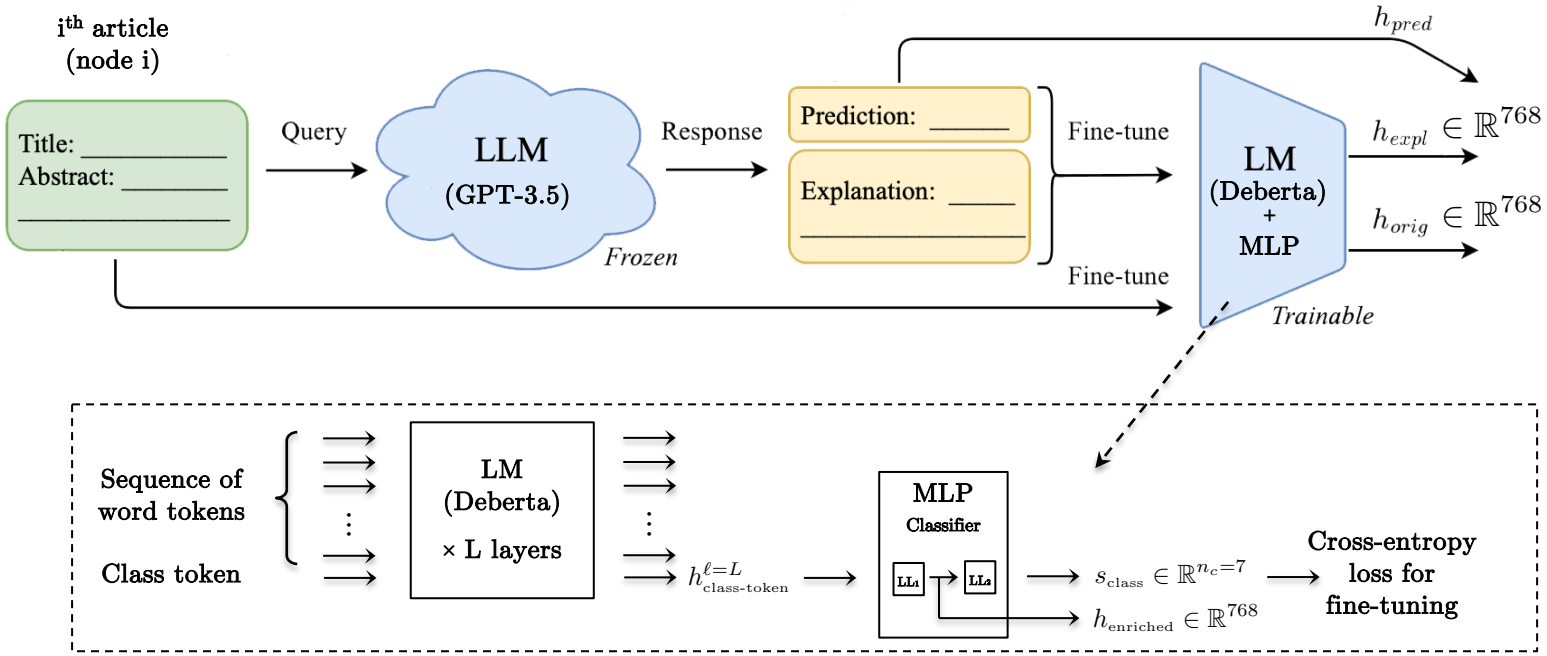



Closed frontier LLMs | Generated text response | Open-weight LLMs | Vector representation

[1] Brown-etal, Language models are few-shot learners, 2020
[2] He-etal, Deberta: Decoding-enhanced bert with disentangled attention, 2021

# TAPE : sequence-to-vector LM interpreter

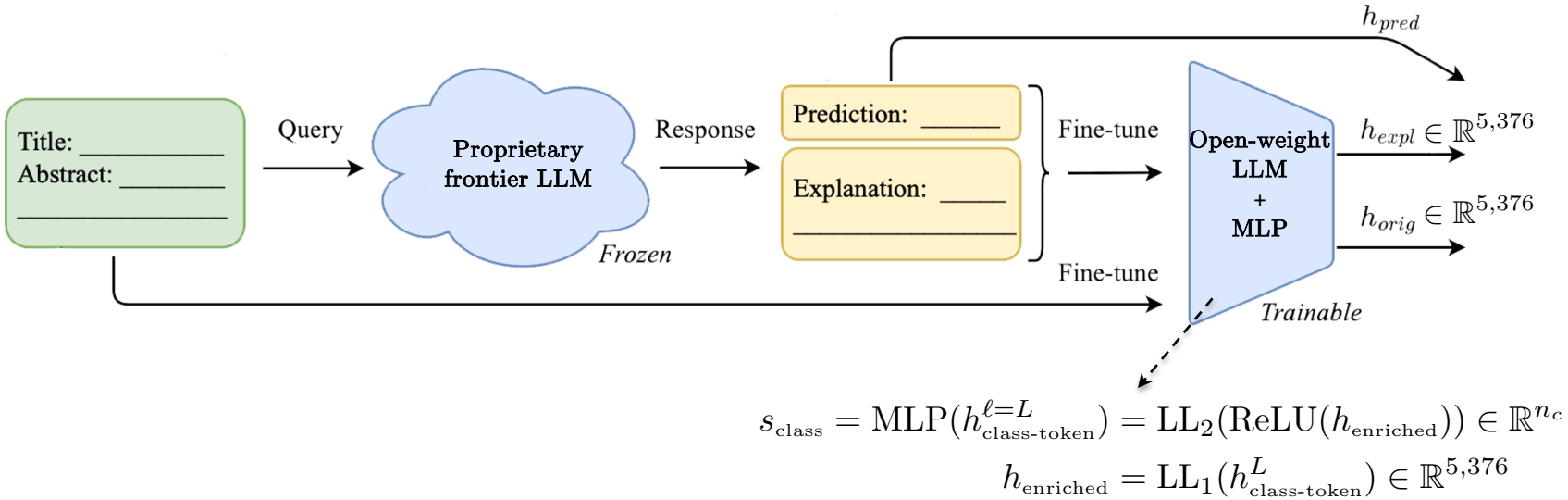
- Pipeline : Node text → prompt → frozen LLM → textual explanation → fine-tunable DeBERTa-base → 768-dim pooled representation $h_i$ → MLP classifier → cross-entropy fine-tuning → discard classifier and freeze $h_i$ as an enriched node feature for GNN training.
- The TAPE interpreter used is a compact DeBERTa-base LM[1] (d = 768, 129M param), rather than an open-weight LLM.




[1] He-etal, Deberta: Decoding-enhanced bert with disentangled attention, 2021

# From LMs to LoRA-adapted open-weight LLMs

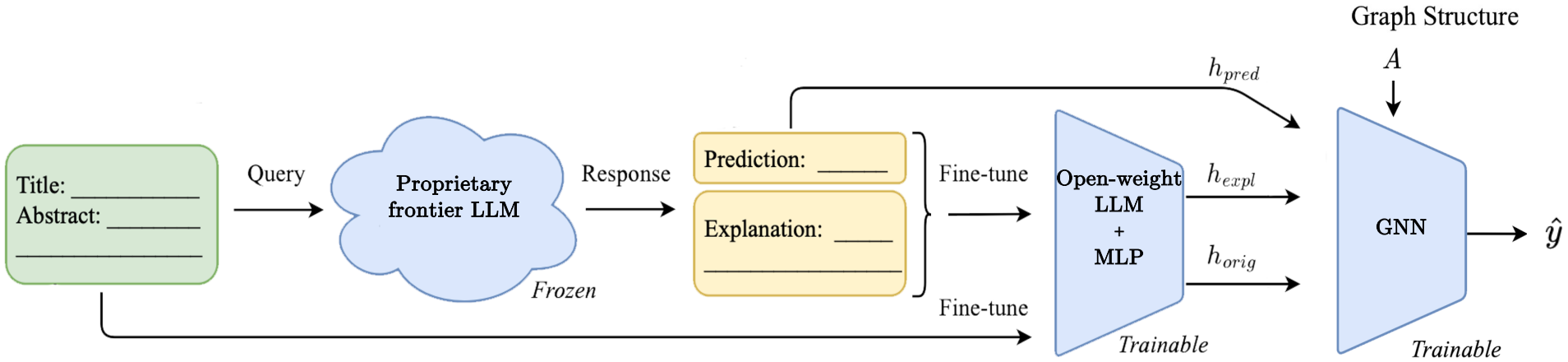
- In 2026, we could replace the small interpreter DeBERTa LM with an open-weight LLM s.a. Gemma 4 31B[1] and adapt it efficiently using LoRA[2].
- Updated pipeline : Node text → prompt → frozen proprietary LLM → prediction + explanation → open-weight LLM with LoRA → pooled hidden representation $h_i \in \mathbb{R}^{5,376}$ → MLP classifier → cross-entropy fine-tuning on the TAG task → discard the classifier and retain the enriched representation $h_i$ as the node feature for downstream GNN training.




[1] Gemma-Team, Gemma 4 Technical Report, 2026
[2] Hu et-al, LoRA: Low-Rank Adaptation of Large Language Models, 2021

# GNN training on enriched features

- Freeze the enriched features and apply standard GNN backbones, here RevGAT[1].
- One GNN is trained per feature type and their predictions are averaged.




$$h_{\text{orig}} \in \mathbb{R}^d \to \text{GNN}_{\text{orig}} \to \hat{y}_{\text{orig}} \in \mathbb{R}^{n_c}$$

$$h_{\text{expl}} \in \mathbb{R}^d \to \text{GNN}_{\text{expl}} \to \hat{y}_{\text{expl}} \in \mathbb{R}^{n_c}$$

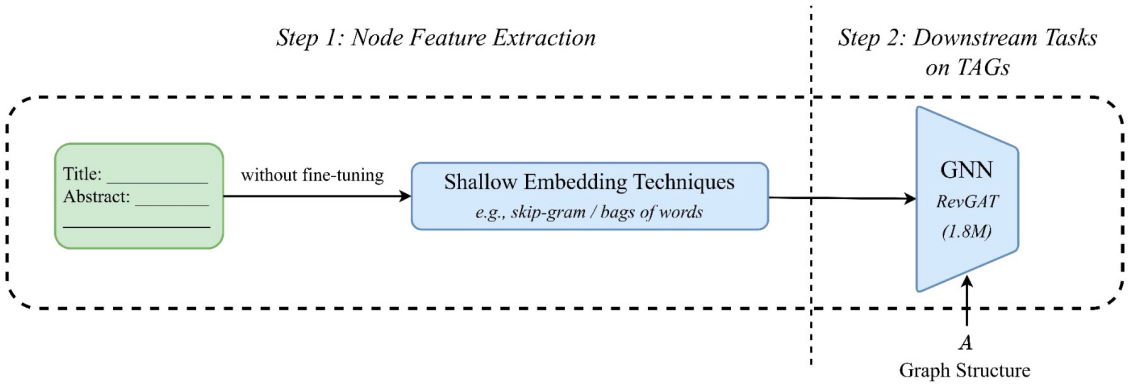
$$h_{\text{pred}} \in \mathbb{R}^{n_c} \to \text{GNN}_{\text{pred}} \to \hat{y}_{\text{pred}} \in \mathbb{R}^{n_c}$$

$$\hat{y} = \frac{\hat{y}_{\text{orig}} + \hat{y}_{\text{expl}} + \hat{y}_{\text{pred}}}{3} \in \mathbb{R}^{n_c}$$

[1] Li-etal, Training graph neural networks with 1000 layers, 2021

# Shallow-feature GNN baseline

- Classic TAG benchmarks are commonly distributed with shallow, text-derived node features.
- For Cora and PubMed, we additionally collected and released the raw title and abstract, which were not available in standard PyG/DGL versions.
- The baseline : Train a GNN directly on the standard shallow text features provided by OGB[1].
- On ogbn-arxiv, RevGAT achieves 70.8% accuracy in 4 min in the TAPE experiment.
- Shallow features therefore provide a fast, inexpensive baseline, but limited semantics.




| Dataset | Methods | Features | Description |
|---|---|---|---|
| Cora | BoW | 1,433 | After stemming and removing stopwords there is a vocabulary of size 1,433 unique words. All words with document frequency less than 10 were removed. |
| PubMed | TF-IDF | 500 | Each publication in the dataset is described by a TF/IDF weighted word vector from a dictionary which consists of 500 unique words. |
| ogbn-arxiv | skip-gram | 128 | The embeddings of individual words are computed by running the skip-gram model (Mikolov et al., 2013) over the MAG (Wang et al., 2020) corpus. |
| ogbn-products | BoW | 100 | Node features are generated by extracting BoW features from the product descriptions followed by a Principal Component Analysis to reduce the dimension to 100. |
| tape-arxiv23 | word2vec | 300 | The embeddings of individual words are computed by running the word2vec model. |

| Dataset | #Nodes | #Edges | Task | Metric | Augmentation |
|---|---|---|---|---|---|
| Cora | 2,708 | 5,429 | 7-class classif. | Accuracy | ✓ |
| Pubmed | 19,717 | 44,338 | 3-class classif. | Accuracy | ✓ |
| ogbn-arxiv | 169,343 | 1,166,243 | 40-class classif. | Accuracy | |
| ogbn-products (subset) | 54,025 | 74,420 | 47-class classif. | Accuracy | |
| tape-arxiv23 | 46,198 | 78,548 | 40-class-classif. | Accuracy | ✓ |

[1] Hu-etal, Open graph benchmark: Datasets for machine learning on graphs, 2020

# GLEM : Alternating LM+GNN training

- GLEM alternates LM and GNN training through a variational EM framework[1].
- On ogbn-arxiv, GLEM achieves 76.6% accuracy in 551 min (9.2 hr) in the TAPE experiment.
- Jointly exploiting textual and graph information improves accuracy, but requires substantially more training time than the shallow-feature GNN baseline.

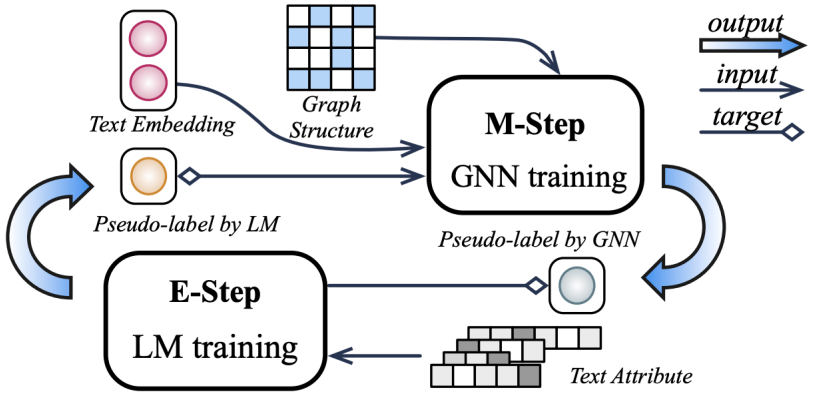
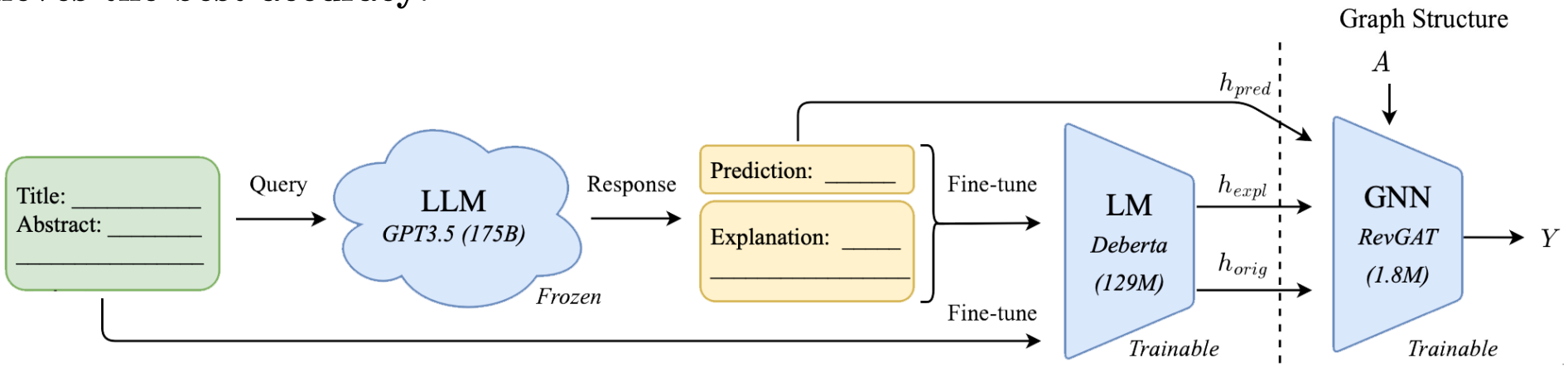



Figure 1: The proposed GLEM framework trains GNN and LM separately in a variational EM framework: In E-step, an LM is trained towards predicting both the gold labels and GNN-predicted pseudo-labels; In M-step, a GNN is trained by predicting both gold labels and LM-inferred pseudo-labels using the embeddings and pseudo-labels predicted by LM.

[1] Zhao-etal, Learning on large-scale text-attributed graphs via variational inference, 2022

# TAPE : LLM-enriched features for GNNs

- Train GNNs using node features enriched with LLM-generated predictions and explanations[1,2].
- On ogbn-arxiv, TAPE achieves 77.5% accuracy with 192 min (3.2 hr) total training time[3].
- Compared with GLEM, TAPE achieves higher accuracy with 2.88× lower training time, showing the benefit of decoupling LLM/LM feature construction from GNN training.
- We also ablate the three sources of TAPE node features on ogbn-arxiv. Removing any feature source reduces performance, combining all three complementary feature sources consistently achieves the best accuracy.

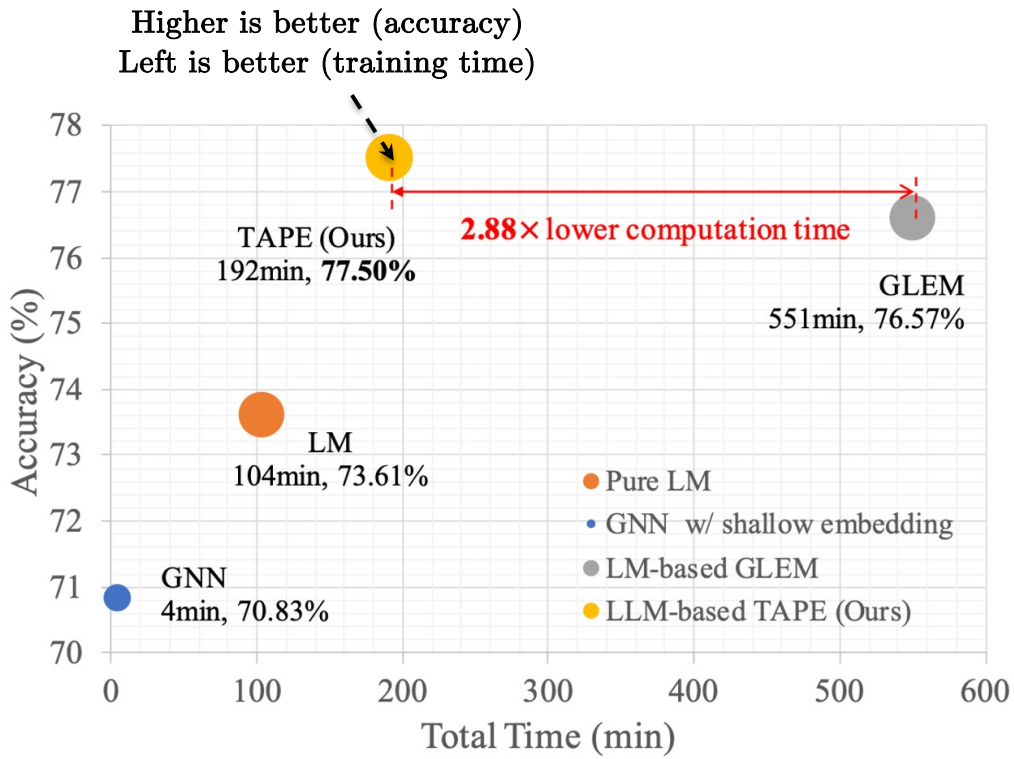




[1] Li-etal, Training graph neural networks with 1000 layers, 2021
[2] Brown-etal, Language models are few-shot learners, 2020
[3] He-LeCun-Bresson-etal, Harnessing Explanations: LLM-to-LM Interpreter for Enhanced Text-Attributed Graph Representation Learning, 2023

# Summary of results

- Text representations strongly affect TAG node-classification performance :
  - Shallow features : Fast, but limited semantic information.
  - LM-based representations : Richer semantics, but substantially higher training cost.
  - LLM-enhanced TAPE features : Best accuracy–efficiency trade-off in this experiment.



# TAPE at the top of the ogbn-arxiv leaderboard

- Snapshot of the OGB ogbn-arxiv node-classification leaderboard (Oct 2026)[1].
- Since 2024, the three highest-ranked submissions have all leveraged TAPE, achieving test accuracies of 78.03%, 77.50%, and 77.48%, respectively.
- These results provide strong empirical evidence that integrating LLM-enriched textual representations with graph neural networks to improve node-classification performance.

Leaderboard for ogbn-arxiv

The classification accuracy on the test and validation sets. The higher, the better.

| Rank | Method | Ext. data | Test Accuracy | Validation Accuracy | Contact | References | #Params | Hardware | Date |
|---|---|---|---|---|---|---|---|---|---|
| 1 | **SimTeG+TAPE+RevGAT** | Yes | 0.7803 ± 0.0007 | 0.7846 ± 0.0004 | Keyu Duan | Paper, Code | 1,386,219,488 | 4 * A100-XMS4 (40GB GPU) | Aug 7, 2023 |
| 2 | **TAPE+RevGAT** | Yes | 0.7750 ± 0.0012 | 0.7785 ± 0.0016 | Xiaoxin He (NUS) | Paper, Code | 280,283,296 | 4 NVIDIA RTX A5000 24GB GPUs | May 31, 2023 |
| 3 | **SimTeG+TAPE+GraphSAGE** | Yes | 0.7748 ± 0.0011 | 0.7789 ± 0.0008 | Keyu Duan | Paper, Code | 1,381,593,403 | 4 * A100-XMS4 (40GB GPU) | Aug 7, 2023 |

[1] https://ogb.stanford.edu/docs/leader_nodeprop/#ogbn-arxiv

# Lab 1 : TAPE

- Boost GCN prediction with LM-enriched features.

## Lecture : Integrating GNNs and LLMs

### Lab 01 : TAPE for boosting GNN predictions with LLM features -- Solution

Xavier Bresson, Zhou Wei

X. He, X. Bresson, T. Laurent, A. Perold, Y. LeCun, B. Hooi, Harnessing Explanations: LLM-to-LM Interpreter for Enhanced Text-Attributed Graph Representation Learning, 2024, https://arxiv.org/pdf/2305.19523

[ ]:
```
# For Google Colaboratory
import sys, os
if "google.colab" in sys.modules:
    # mount google drive
    from google.colab import drive
    drive.mount('/content/gdrive')
    path_to_file = '/content/gdrive/My Drive/CS5284_2026_codes/08_LLMs_GNNs'
    print(path_to_file)
    # change current path to the folder containing "path_to_file"
    os.chdir(path_to_file)
    !pwd
    !pip -q install -f https://data.dgl.ai/wheels/torch-2.6/repo.html dgl==2.5.0 # Install DGL
```

### GNN + LLM text embeddings for Cora Classification

In TAPE, large language model information is used to enrich node representations before graph learning.

This lab studies a simpler version of TAPE with the pipeline :

```
Cora dataset
        |
        +--> standard BoW node features --> GCN
        |
        +--> title + abstract --> pre-trained LM --> enhanced node features --> GCN
```

### Compare standard GCN and GCN augmented with LM features

[11]:
```
standard_row = {}
standard_row["features"] = "Model 1: GCN on Cora standard BoW"
standard_row["dim"] = x_standard.shape[1]
standard_row["best_val_acc"] = result_standard["best_val_acc"]
standard_row["test_acc"] = result_standard["test_acc"]
standard_row["gcn_train_sec"] = result_standard["train_seconds"]

lm_row = {}
lm_row["features"] = f"Model 2: SmolLM2-360M on title+abstract"
lm_row["dim"] = x_lm.shape[1]
lm_row["best_val_acc"] = result_lm["best_val_acc"]
lm_row["test_acc"] = result_lm["test_acc"]
lm_row["gcn_train_sec"] = result_lm["train_seconds"]

results = pd.DataFrame([standard_row, lm_row])
results
```

[11]:

| | features | dim | best_val_acc | test_acc | gcn_train_sec |
|---|---|---|---|---|---|
| 0 | Model 1: GCN on Cora standard BoW | 1433 | 0.803324 | 0.79631 | 2.559628 |
| 1 | Model 2: SmolLM2-360M on title+abstract | 960 | 0.835642 | 0.83321 | 2.120933 |

# Outline

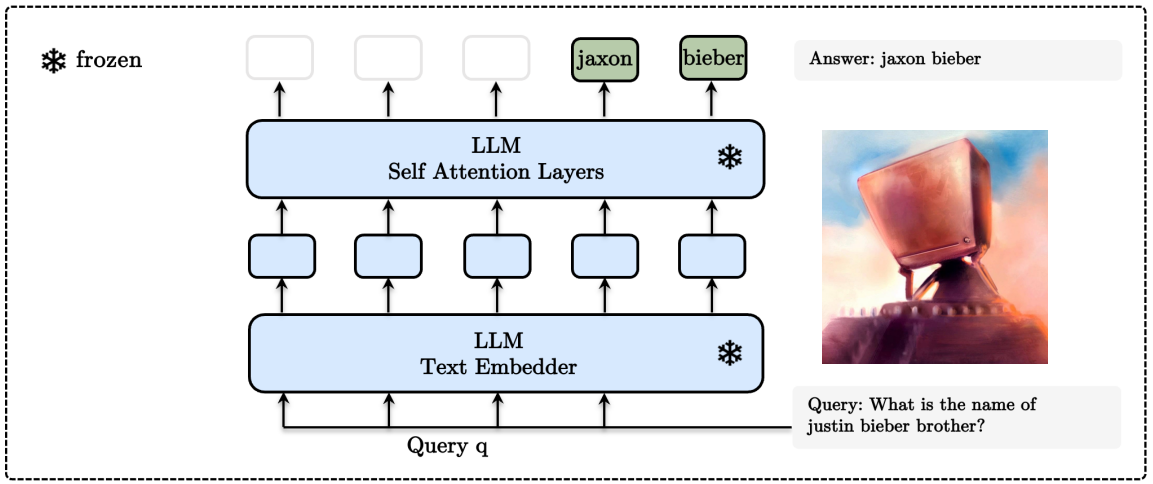
- LLMs and GenAI
- Do LLMs make GNNs obsolete ?
- Combining LLMs and GNNs
- Using LLM for GNN reasoning
- **Using GNN for LLM reasoning**
- Conclusion

# Why zero-shot QA may not be enough ?

- Question answering (QA) task evaluates many NLP capabilities.
- We will improve QA with progressively richer evidence.
- Approach 1 / zero-shot : Prompt the LLM using the question alone, then let it generate an answer without task-specific training.
- Advantage : Simply use English with no model re-training.
- Limitation : Plausible generated answers can be factually incorrect, a.k.a. hallucinations. High token probability does not guarantee truth.

Proprietary LLMs

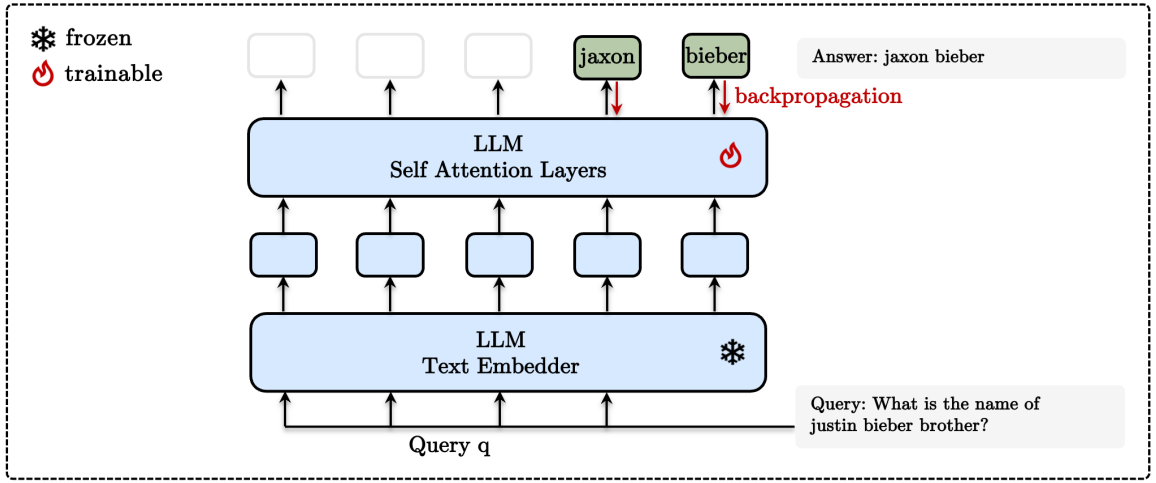
Open-weight LLMs



# Fine-tuning LLM for QA

- Approach 2 : Fine-tuning on labeled QA data, select hyper-parameters on validation data and evaluate on a test set.
- Benefit : Task adaptation can improve accuracy and answer consistency, although it does not eliminate the risk of hallucinations.
- Limitation : Proprietary models may allow only API-based fine-tuning. For large open-weight models, updating all weights requires substantial GPU memory, compute and high-quality training data.

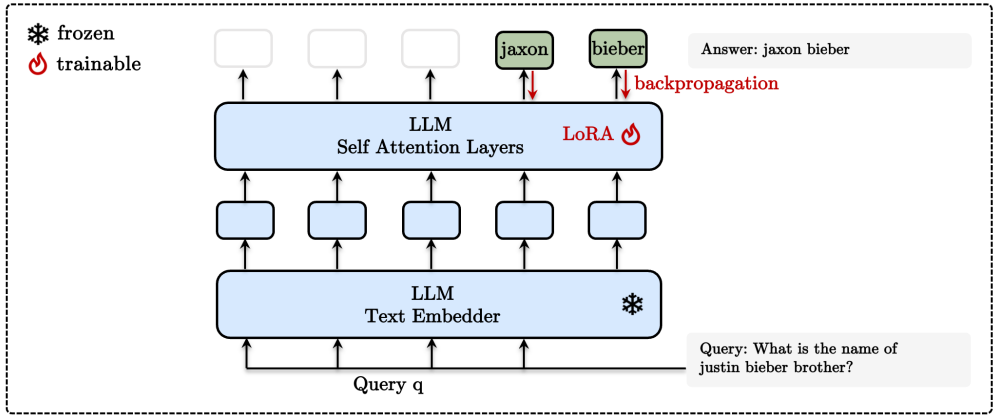
Fine-tuning open-weight LLM



# LoRA : Parameter-efficient QA fine-tuning

- LoRA[1] trains low-rank weight updates while freezing the base model, often updating much less than 1% of the original parameters.
- In 2026, models range from Gemma 4 31B to Muse Spark Open 816B and DeepSeek-V4-Pro 1.6T.
- Full tuning stores model weights, gradients and optimizer states. Its GPU requirements can range from 8-12 × H100 for Gemma 4 to 400 × H100 for DeepSeek-V4 Pro, depending on precision and parallelization settings.
- LoRA reduces significantly trainable weights, but not the base-model memory footprint. In 2024, for Llama 2 7B with rank = 8 and $\alpha = 16$, we trained 4.2M weights (0.06% of LLM weights) using 1 × A5000 (academic server).

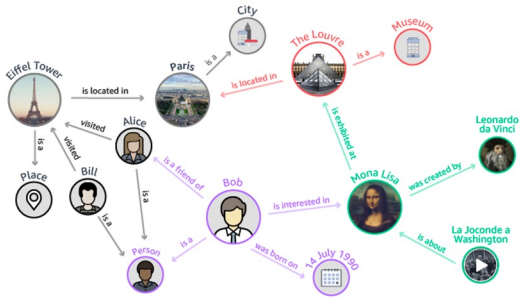
LoRA fine-tuning
with open-weight LLM
Llama 2 7B



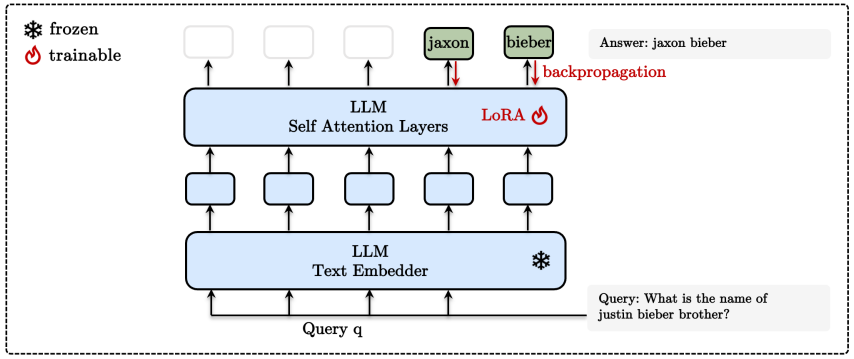

[1] Hu-etal, LoRA: Low-Rank Adaptation of Large Language Models, 2021

# Grounding QA in knowledge graph

- Goal[1] : Improve factual QA by grounding answers in retrieved graph-structured data, rather than relying only on the LLM's internal knowledge.
- A knowledge graph represents entities and their relations, supplying evidence for questions that require multiple hops. This extends the RAG[2] idea of retrieving context before generation.
- Challenge : Retrieve a query-relevant knowledge graph and encode it so that the LLM can exploit both its textual and structural information.




Knowledge graph containing evidence relevant to the user's query

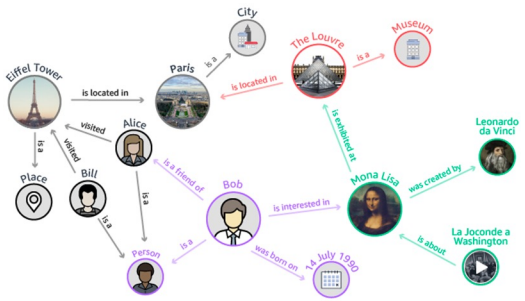



How can the LLM exploit the retrieved graph ?

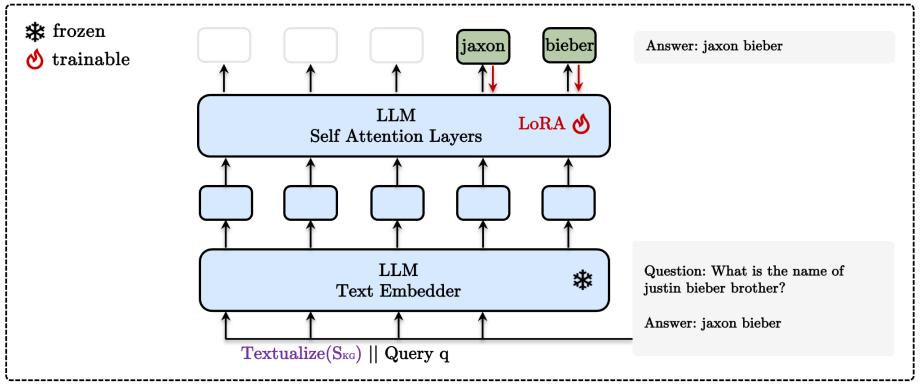

[1] He-LeCun-Bresson-etal, G-retriever: Retrieval-augmented generation for textual graph understanding and question answering, 2024
[2] Lewis-etal, Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, 2020

# Making graph evidence readable to LLMs

- Text-only LLMs read token sequences rather than graph-structured objects.
- We therefore serialize selected node labels, edges and attributes into text, then concatenate that text with the user question/query.
- The textualized graph provides explicit and interpretable language semantic evidence for answer generation.



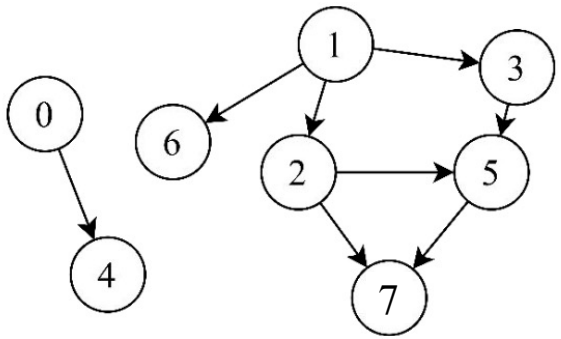

Knowledge graph $S_{KG}$




Add the textualized knowledge graph $S_{KG}$ to the user query

# Graph-to-Text : Serializing nodes and edges

- Graph textualization writes node IDs, attributes and relations in a chosen sequence that a language model can process.
- Example of graph textualization : A directed edge (i,j) indicates that node i points to node j. Its textual serialization preserves both the edge direction and the associated relation label.




Graph G

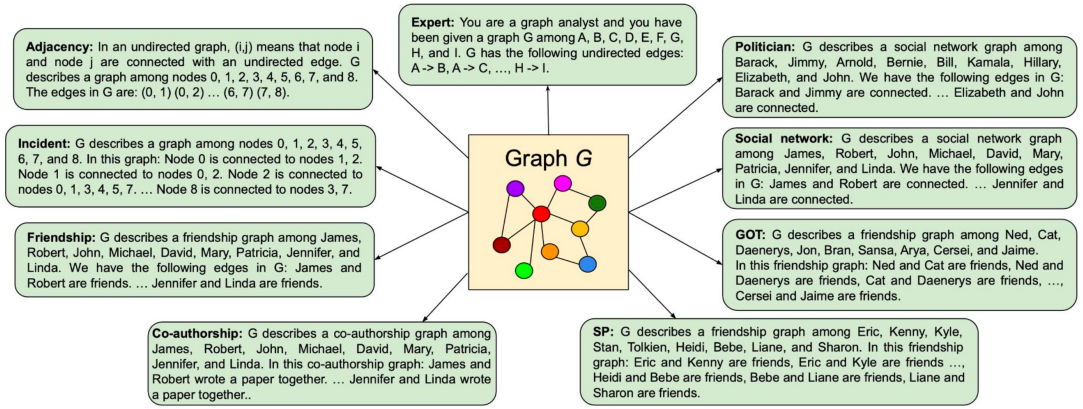
$\Leftrightarrow$

A graph G is a set of directed edges defined by (i,j) where node i points to node j. The graph G is defined with edges (0,4), (1,3), (1,2), (1,6), (2,5), (2,7), (3,5), (5,7).

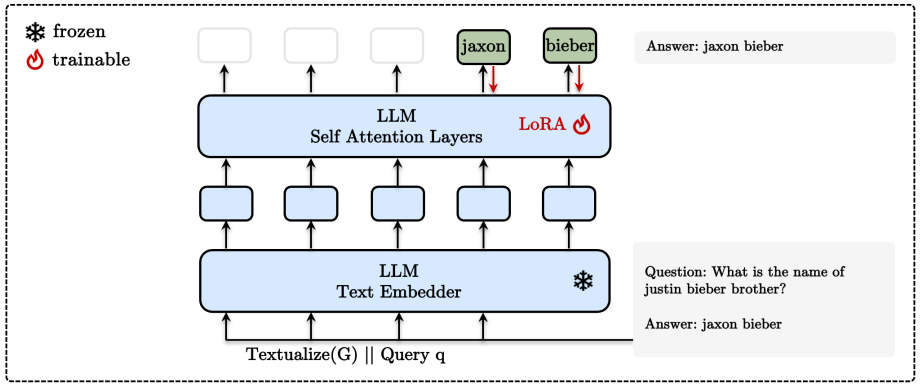
Textualize(G)

# Graph textualization limitations

- Non-uniqueness : Many token sequences can describe the same graph, depending on node IDs, relation labels and traversal order.
- No text-invariance : Equivalent serializations can provide different LLM outputs or graph-reasoning accuracy despite identical graph structure[1]. Hence, LLMs are not invariant to these representations.




Different encoding strategies from graph G to text[1]

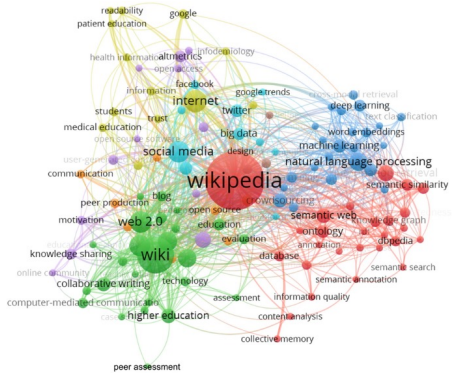



Add the textualized knowledge graph G to the open-weight LLM

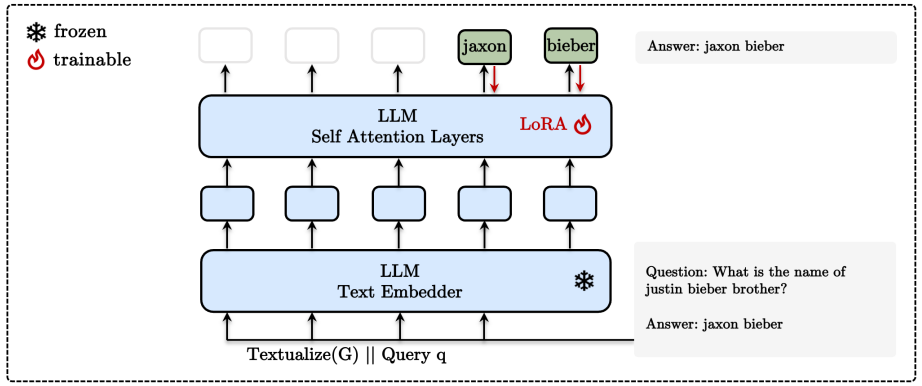
[1] Fatemi-etal, Talk like a graph: Encoding graphs for large language models, 2024

# Graph textualization does not scale

- Serializing all graph nodes, links and text can create an input much longer than a language model's context window.
- Open-weight LLMs have finite context windows : In 2024, we used Llama 2 7B[1] with a 4,096-token context. In Oct 2026, Muse Spark Open[2] and DeepSeek-V4-Pro[3] support 1M tokens.
- Real-world graphs exceed these limits : English Wikipedia from Jan 2025 contains 6.74M articles nodes and 159.6M directed links[4].
- Even million-token windows cannot hold this whole network, making graph retrieval essential.




Wikipedia network[5] G

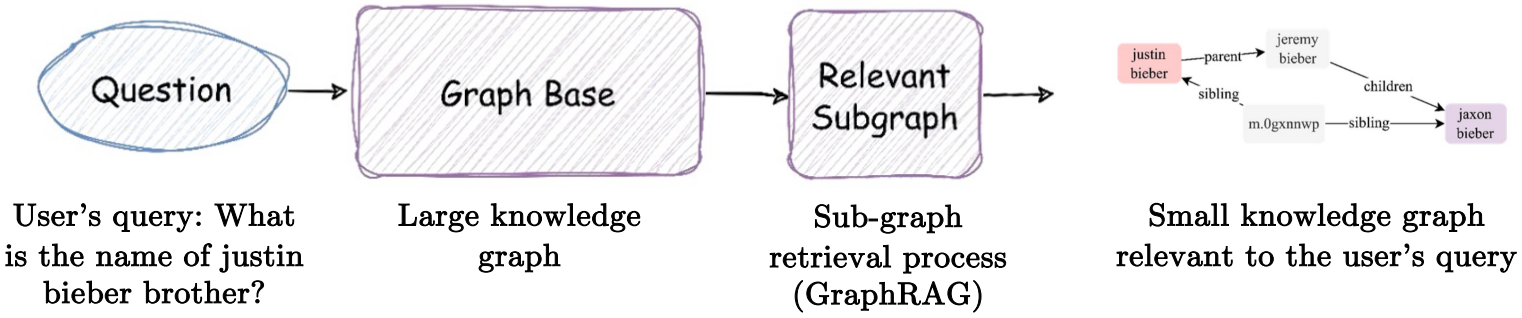



As of 2026, incorporating the whole Wikipedia into an open-weight LLM remains impossible.


[1] Touvron-etal, Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023
[2] Model card 2026, https://huggingface.co/meta-models/Muse-Spark-Open-816B-A42B-NVFP4
[3] Xu-etal, DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence, 2026
[4] Agapitos-etal, WikiTextGraph: A Python Tool for Parsing Multilingual Wikipedia Text and Graph Extraction, 2025
[5] Mostafa, Twenty years of Wikipedia in scholarly publications: a bibliometric network analysis of the thematic and citation landscape, 2023

# GraphRAG : Retrieve a sub-graph before generation

- We adopt a simple graph-to-text serialization approach for its simplicity and interpretability.
- However, serializing large graphs is impractical. Instead, we retrieve a query-relevant portion of the original graph to ground the LLM with structured evidence.
- This retrieval process produces a compact, connected sub-graph, avoiding the need to provide the entire graph to the LLM.
- This approach is called GraphRAG[1], by analogy with standard RAG, which grounds LLM responses in query-relevant retrieved documents.
- Our GraphRAG pipeline consists of three stages (detailed next) : Graph indexing, candidate retrieval and PCST-based sub-graph selection.




User's query: What is the name of justin bieber brother?

Large knowledge graph

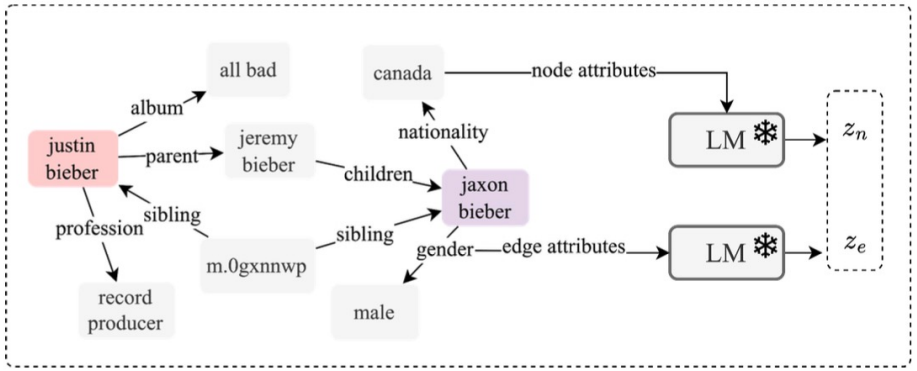
Sub-graph retrieval process (GraphRAG)

Small knowledge graph relevant to the user's query

[1] The term GraphRAG was introduced by the NebulaGraph team (Vesoft Inc.) in 2023.

# Step 1 : Index textual nodes and edges

- Large graph embedding : Encode all textual attributes of nodes and edges using a frozen text encoder (here a LM Sentence-BERT-large[1] with 355M parameters), producing semantic embeddings $z_n$, $z_e \in \mathbb{R}^d$.
- Graph and vector indexing : Store embeddings in a vector index for efficient similarity search, while preserving graph connectivity in a graph database for structure-aware retrieval[1-6].

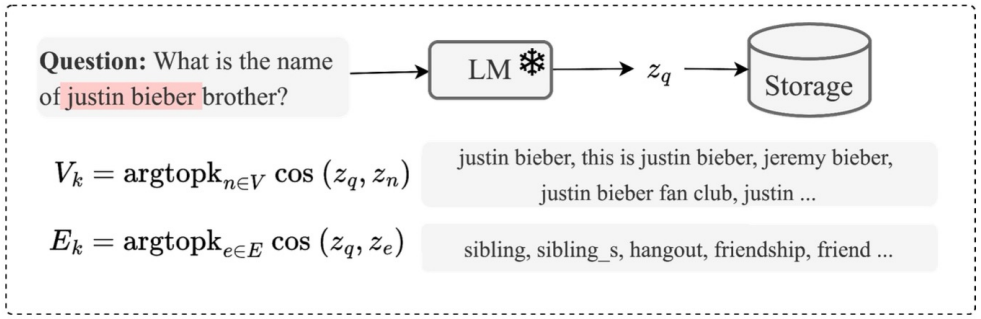




[1] Reimers-Gurevych, Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, 2019
[2] PyTorch Geometric (PyG), https://pyg.org
[3] Deep Graph Library (DGL), https://www.dgl.ai
[4] Llama-index, https://www.llamaindex.ai
[5] Microsoft GraphRAG, https://microsoft.github.io/graphrag
[6] NebulaGraph, https://www.nebula-graph.io
[7] Neo4j, https://neo4j.com

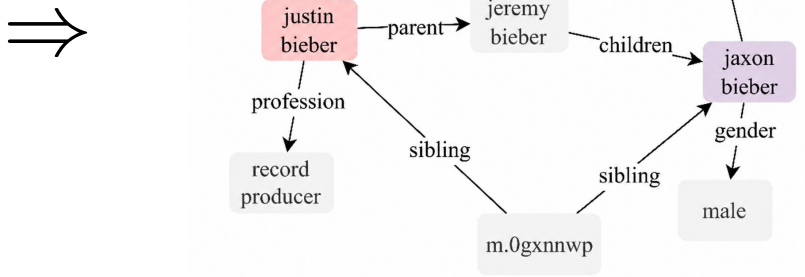
# Step 2 : Retrieve query-relevant candidates

- Encode the question q with the same model : $z_q = \mathrm{LM}(q) \in \mathbb{R}^d$.
- Rank node and edge embeddings separately by cosine similarity to $z_q$, then select the top-k elements in each group.
- The candidate graph may contain noise or disconnected elements. The next step selects compact, connected graph evidence.




$$\text{cos-sim}(z_i, z_j) = \frac{z_i^T z_j}{\|z_i\| \|z_j\|} \in [-1, 1], \; z_i, z_j \in \mathbb{R}^d$$

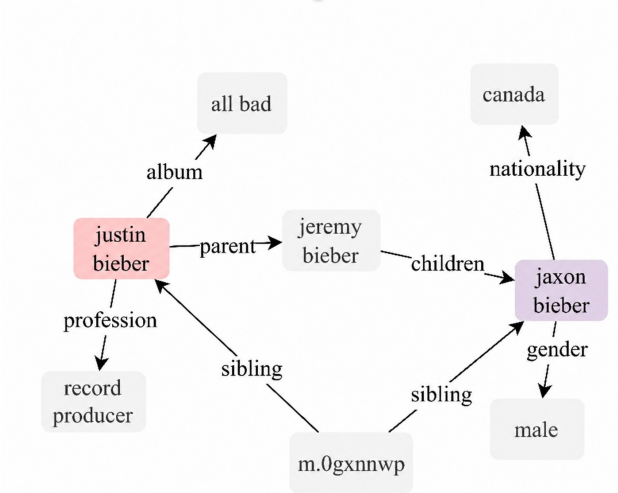
$$\text{with } z_k = \mathrm{LM}(y_k) \in \mathbb{R}^d, \; k = \{i, j\}$$




$$G = (V, E)$$

# Step 3 : Select a compact connected sub-graph

- Construct a compact connected sub-graph S* that balances relevance evidence against graph size.
- G-Retriever gives relevance prizes to nodes and edges, then uses Prize-Collecting Steiner Tree (PCST) to trade off those prizes against the cost of selected edges[1].



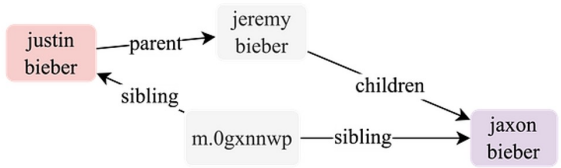

Original graph

$G = (V, E)$

**Prize-Collecting Steiner Tree Optimization**

$$S^* = \underset{\substack{S \subseteq G \\ S \text{ connected}}}{\arg\max} \sum_{n \in V, e \in E} \text{prize}(n, e) - \text{cost}(S)$$

$\Rightarrow$




Prize-collecting Steiner tree

$S^* = (V^*, E^*)$

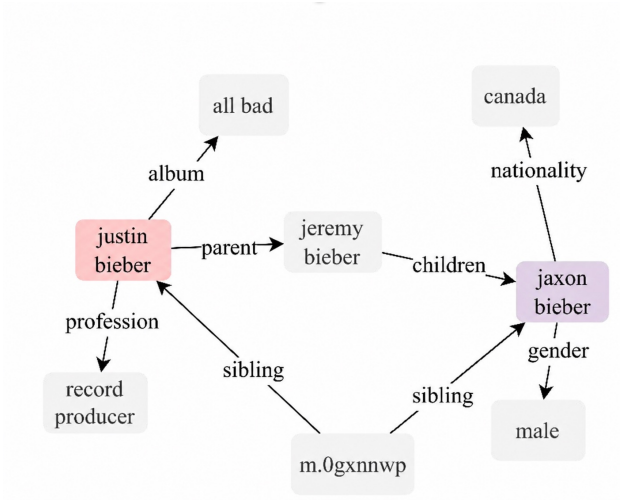
[1] Bienstock et-al, A note on the prize collecting traveling salesman problem, 1993

# Assigning query-relevance prizes for PCST

- **We assign rank-based prizes to the top-k retrieved nodes and edges :**

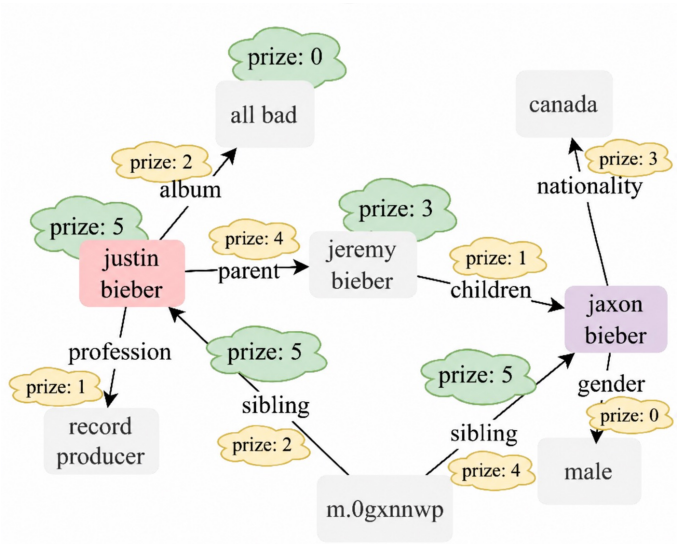
$$\text{prize}(n) = k - i, \text{ if } n \in V_k \text{ and } n \text{ is the top } i \text{ node, else } 0$$

$$\text{prize}(e) = k - i, \text{ if } e \in E_k \text{ and } e \text{ is the top } i \text{ edge, else } 0$$




Original graph
$G = (V, E)$

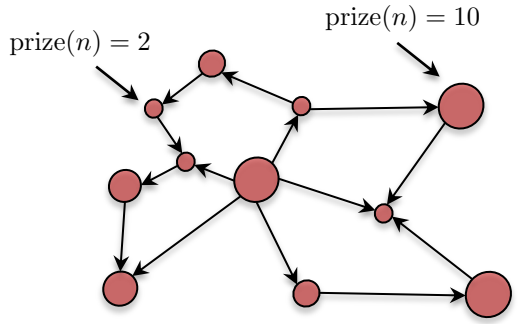
$\Rightarrow$




$\text{prize}(n) =$ green bubbles
$\text{prize}(e) =$ yellow bubbles

# PCST : Relevance vs. sub-graph size

- Prize-Collecting Steiner Tree seeks a connected sub-graph that maximizes selected node prizes minus a chosen fixed edge costs $\lambda > 0$. High-prize nodes outweigh the cost of the edges needed to connect them, i.e. price(n) $\gg \lambda$.
- PCST is originally designed for undirected graphs. G-Retriever preserves the original directed relations, although the retrieved subgraph may not be strongly connected, e.g. $n_a \rightarrow n_b \leftarrow n_c$.
- PCST is also NP-hard, but efficient approximation methods exist. G-Retriever uses a near-linear-time subgraph-selection routine[1,2].

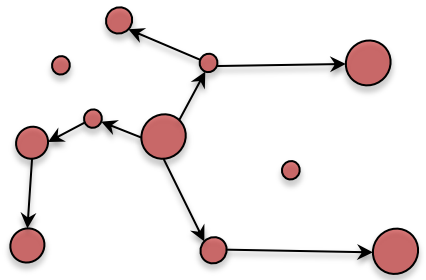



Original graph
$G = (V, E)$

$$S^* = \underset{\substack{S \subseteq G \\ S \text{ connected}}}{\arg\max} \sum_{n \in V} \text{prize}(n) - \text{cost}(S),$$

$$\text{with cost}(S) = \lambda |E_S|, \lambda > 0$$

$$\Rightarrow$$



Prize-collecting Steiner tree
$S^* = (V^*, E^*)$

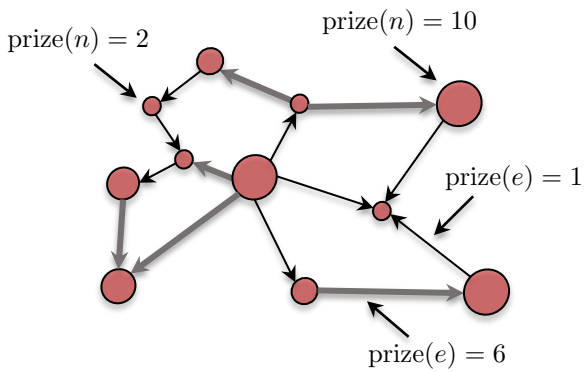
[1] Hegde-etal, A Nearly-Linear Time Framework for Graph-Structured Sparsity, 2015
[2] Goemans-Williamson, A General Approximation Technique for Constrained Forest Problems, 1995

# Extending PCST with edge prizes

- Standard PCST supports node prizes and edge costs, but does not directly assign positive prizes to selected edges.
- G-Retriever rewards textually relevant relations as well as nodes, making edge evidence part of the retrieval objective.
- If price(e) ≤ λ, the edge cost becomes λ−price(e). If price(e) > λ, the edge is replaced by a virtual node $n_v$ s.t. e = ($n_a$, $n_b$) ⇒ $n_a \rightarrow n_v \rightarrow n_b$, with prize($n_v$) = price(e)−λ, allowing the same near-linear-time PCST solver to be used[1].

$$S^* = \underset{\substack{S \subseteq G \\ S \text{ connected}}}{\arg\max} \sum_{n \in V} \text{prize}(n) + \sum_{e \in E} \text{prize}(e) - \text{cost}(S),$$

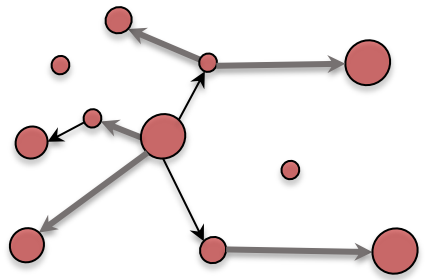
$$\text{with cost}(S) = \lambda |E_S|, \lambda > 0$$




Original graph

$G = (V, E)$

⇒



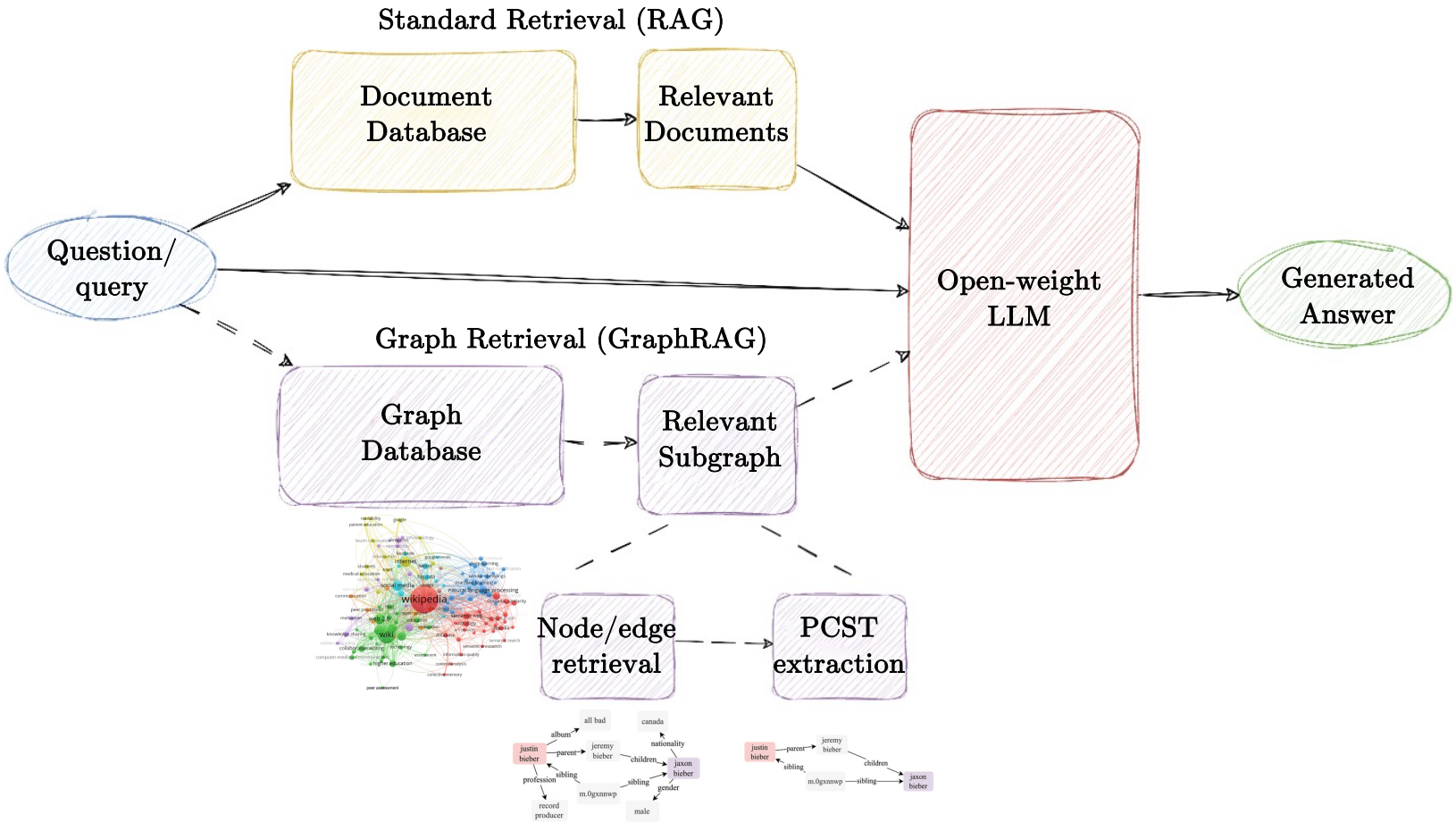
Prize-collecting Steiner tree

$S^* = (V^*, E^*)$

[1] Hegde-etal, A Nearly-Linear Time Framework for Graph-Structured Sparsity, 2015

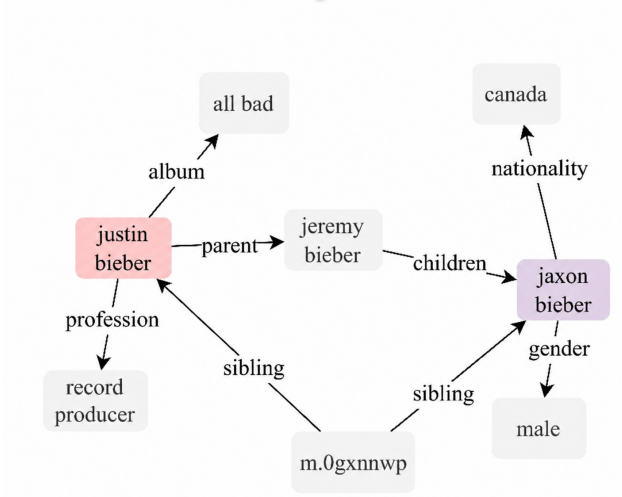
# From document retrieval to graph retrieval

- Standard RAG primarily retrieves semantically relevant text chunks.
- GraphRAG ranks graph nodes and relations, then uses PCST to select connected evidence before the LLM generates an answer.

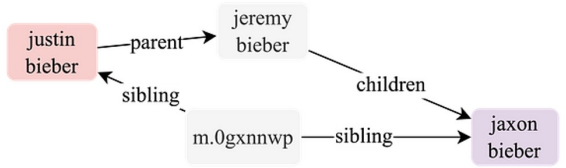


# A simple subgraph-to-text format

- **Serialize each node and directed relation as a compact text row :**




Original graph
$G = (V, E)$




Prize-collecting Steiner tree
$S^* = (V^*, E^*)$

```
node_id, node_attr
15, justin bieber
294, jaxon bieber
356, jeremy bieber
551, m.Ogxnnwp
src, edge_attr, dst
294, parents, 356
356, children, 15
551, sibling, 294
551, sibling, 15
```

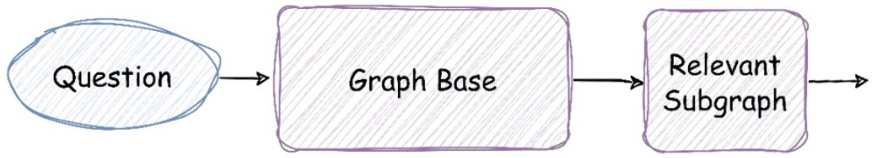
nodes of S*

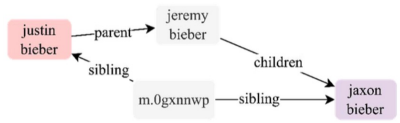
edges of S*

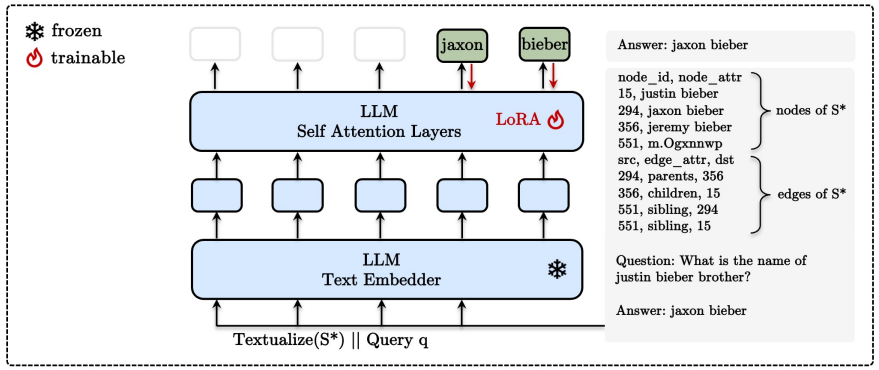
Textualize(S*)

# Condition the LLM on the textualized graph

- After finding S*, serialize its nodes and edges into text, then combine that evidence with the user question.
- Scalability : Only selected relations use prompt tokens.
- Grounding : Retrieved facts can support more relevant responses, mitigating hallucinations.




User's query: What is the name of justin bieber brother?

Large knowledge graph

Sub-graph retrieval process (GraphRAG)

Small knowledge graph S* relevant to the user's query

Add the textualized knowledge graph S* to the open-weight LLM

# Two ways to represent the retrieved graph

- G-Retriever is a GraphRAG method that provides the LLM with retrieved graph evidence in two complementary forms :
  - Graph-to-text : Textualize node and edge attributes from the retrieved subgraph, then concatenate the resulting text with the user query.
  - Graph token (presented next slide) : Represent the retrieved sub-graph as a learned graph embedding for the LLM.
- Graph text supplies explicit facts, while the graph token summarizes structural context useful for answer generation.

# Building a graph soft-prompt token

- A graph soft prompt represents the retrieved graph as an LLM vector[1].
- Apply a GNN / Graph Transformer to contextualize node features.
- Mean-pool final node states into a graph representation $h_{graph} \in \mathbb{R}^{d_{GNN}}$.
- Project $h_{graph}$ with an MLP into the LLM embedding dimension $h_{gra\text{-}tok} \in \mathbb{R}^{d_{LLM}}$.
- Concatenate the projected vector $h_{gra\text{-}tok}$ as a continuous graph prompt token.
- Train the GNN and projection via the downstream QA loss.

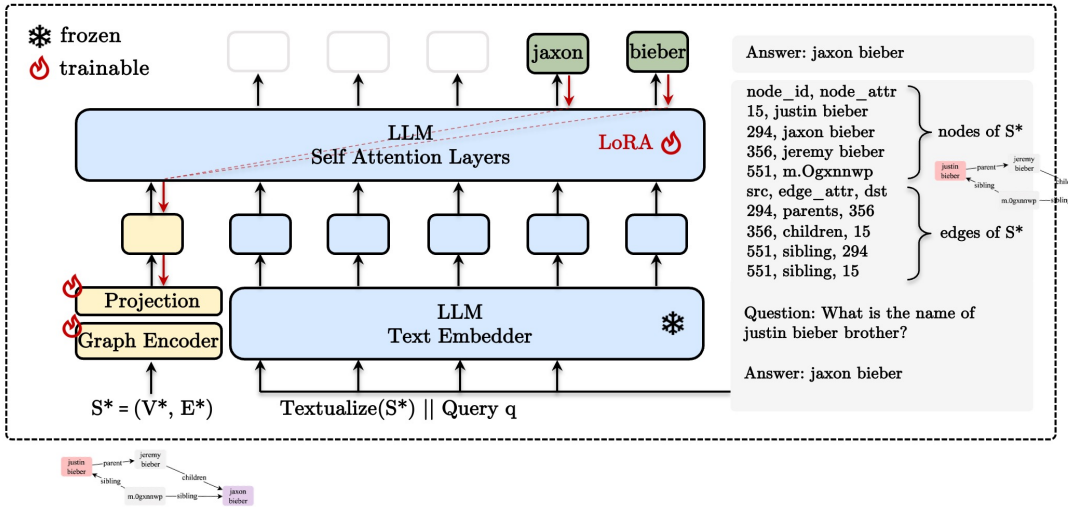
For $\ell = 0, ..., L-1$

$$h_i^{\ell+1} = f_{\text{node}}(h_i^\ell, \{h_j^\ell\}_{j\in\mathcal{N}_i}, \{e_{ij}^\ell\}_{j\in\mathcal{N}_i}) \in \mathbb{R}^{d_{\text{GNN}}}$$

$$e_{ij}^{\ell+1} = f_{\text{edge}}(e_{ij}^\ell, h_i^\ell, h_j^\ell) \in \mathbb{R}^{d_{\text{GNN}}}$$

$$h_{\text{graph}} = \text{mean}_{i\in V^*}(h_i^L) \in \mathbb{R}^{d_{\text{GNN}}}$$

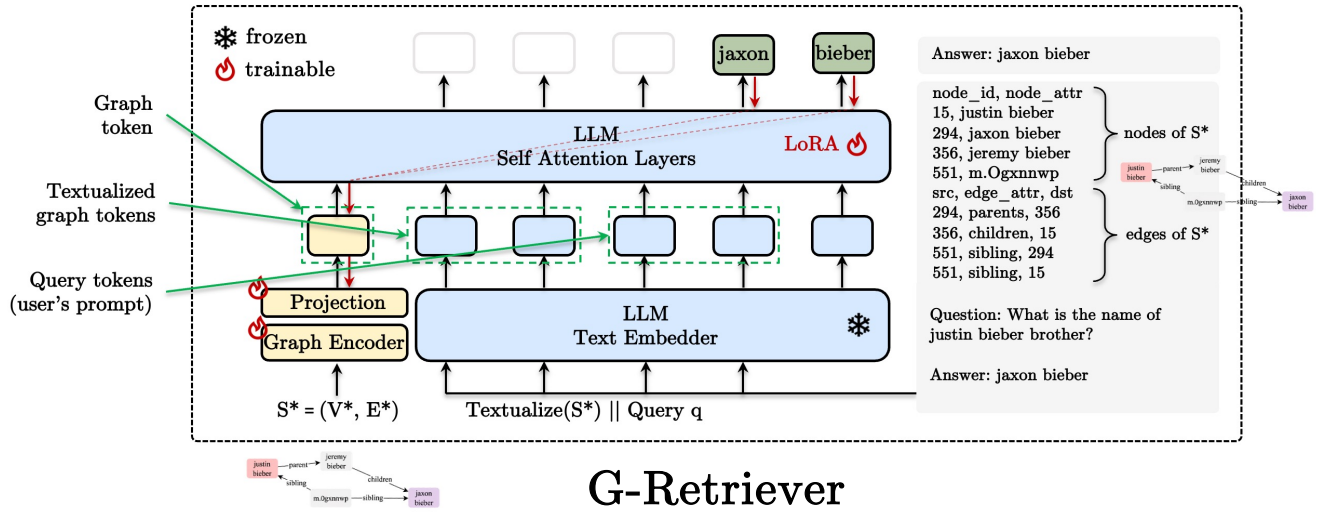
$$h_{\text{gra-tok}} = \text{MLP}(h_{\text{graph}}) \in \mathbb{R}^{d_{\text{LLM}}}$$




[1] Chai-etal, Boosting Graph Reasoning Ability of Large Language Model, 2023

# The full G-Retriever pipeline

- Step 1 -- Index : Embed graph-node and edge text in advance, while keeping connectivity and relation metadata accessible.
- Step 2 – Retrieve : Rank candidates against the user question, then extract a compact connected sub-graph using PCST.
- Step 3 -- Encode : Textualize S* and build a projected graph token, using a graph encoder and a small MLP.
- Step 4 – Generate : Condition the LLM on graph text, token and query.
- Step 5 -- Train : Next slide




G-Retriever

# How G-Retriever is trained

- Forward pass : Combine the graph token, retrieved graph text and user question. Use teacher forcing on the answer prefix.
- Objective : Minimize cross-entropy over target answer tokens.

$$\mathcal{L}_{\mathrm{CE}}(\theta) = -\sum_{i=1}^{L_X} \log p_\theta \left( y_i \mid y_{<i}, h_{\mathrm{graph}}, y_{\mathrm{GraphRAG}}, q \right)$$

- Backward pass : Train the graph encoder/projection and LoRA adapters, while keeping the base LLM text embedder frozen.
- Open-weight LLM in 2024 : Llama 2 7B with LoRA r = 8, $\alpha = 16$ (4.2M parameters) and a four-layer Graph Transformer with four attention heads and hidden 1024-dimension (21M parameters), plus projection MLP (10.5M).

# GraphQA Results : Accuracy, efficiency, faithfulness

- We introduced GraphQA[1], which covers three domains : ExplaGraphs (commonsense), SceneGraphs (scene understanding) and WebQSP (knowledge QA).
- Under the reported settings, G-Retriever outperforms the compared graph-prompt and LoRA baselines on all three datasets.
- On WebQSP, retrieval cuts graph-text tokens by about 99% and training minutes per epoch by about 67%.
- On SceneGraphs, fully valid output graphs rise from 8% to 62%, a gain of 54% over graph prompt tuning.
- WebQSP ablations show Hit@1 drops of 22.5% without GNN and 19.2% without graph text (9.4% without retrieval). Takeaway : Explicit graph evidence and GNN features complement.

| Dataset | ExplaGraphs | SceneGraphs | WebQSP |
|---|---|---|---|
| #Graphs | 2,766 | 100,000 | 4,737 |
| Avg. #Nodes | 5.17 | 19.13 | 1370.89 |
| Avg. #Edges | 4.25 | 68.44 | 4252.37 |
| Node Attribute | Commonsense concepts | Object attributes (*e.g.,* color, shape, material, activity) | Entities in Freebase |
| Edge Attribute | Commonsense relations | Relations (*e.g.,* actions, spatial relations, comparatives) | Relations in Freebase |
| Task | Common sense reasoning | Scene graph question answering | Knowledge based question answering |
| Evaluation Matrix | Accuracy | Accuracy | Hit@1 |

[1] https://github.com/XiaoxinHe/G-Retriever/tree/main/dataset

# Lab 2 : G-Retriever

- Implement G-Retriever and GraphRAG.

## Lecture : Integrating GNNs and LLMs

### Lab 02 : G-Retriever with LLMs, GNNs and GraphRAG -- Solution

Xavier Bresson, Ryoji Kubo

He, Tian, Sun, Chawla, Laurent, LeCun, Bresson, Hooi, G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering, 2024, https://arxiv.org/abs/2402.07630

[1]:
```
# For Google Colaboratory
# Go to Runtime → Change runtime type → Select T4 GPU to accelerate LM embedding computation
import sys, os
if "google.colab" in sys.modules:
    # mount google drive
    from google.colab import drive
    drive.mount('/content/gdrive')
    path_to_file = '/content/gdrive/My Drive/CS5284_2026_codes/08_LLMs_GNNs'
    print(path_to_file)
    # change current path to the folder containing "path_to_file"
    os.chdir(path_to_file)
    !pwd
    !pip -q install -f https://data.dgl.ai/wheels/torch-2.6/repo.html dgl==2.5.0 # Install DGL
    # Install pcst_fast
    import importlib.util, subprocess, sys
    if importlib.util.find_spec('pcst_fast') is None:
      subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'pybind11>=2.13', 'setuptools_scm', 'wheel'])
      subprocess.check_call([sys.executable, '-m', 'pip', 'install', '--no-build-isolation', '--no-deps', '--no-binary=pcst-fast', 'pcst-
```

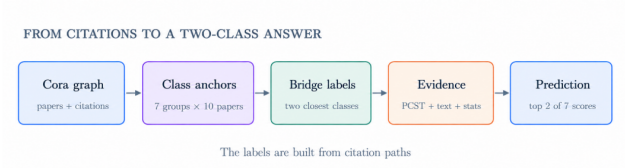
### G-Retriever on Cora : Which two research classes does a paper bridge ?

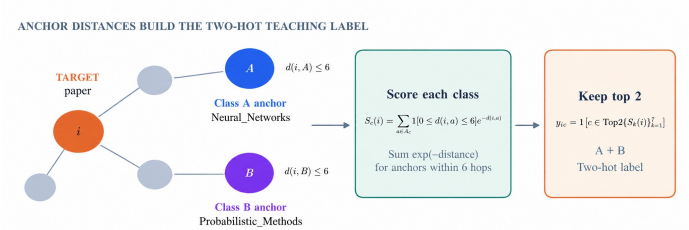
**Learning goal:** Given a *TARGET* paper in the Cora citation network, predict *two research classes* that are structurally close to it. This is a fundamentally a graph-based question, not the original single Cora classification label task. For example, a paper connected by short citation paths to both `Neural_Networks` and `Probabilistic_Methods` classes is labeled as bridging those two classes.

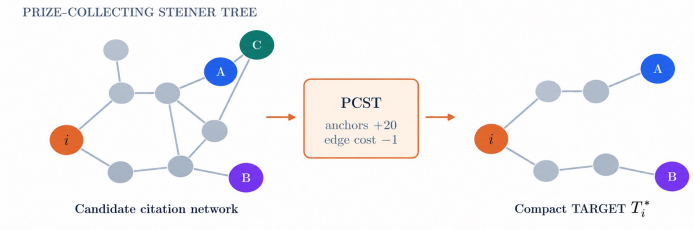
Training set: We construct teaching labels (here the two classes that are the closest to the target paper) from shortest paths and a few paper anchors per class.

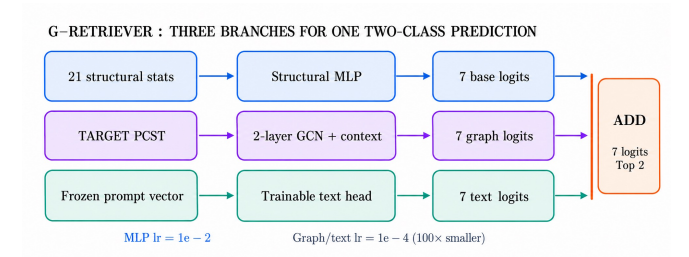
Workflow: We load the CORA citation graph → encode paper text → choose labeled class anchors and build two-class targets → retrieve an evidence subgraph with PCST → compare a standard full-graph GCN with sparse GraphRAG.

In this notebook, the retrieved PCST evidence is defined as the query-specific citation subgraph, including its selected papers and edges, connected class anchors, paths within the retrieved subgraph and neighboring paper titles.



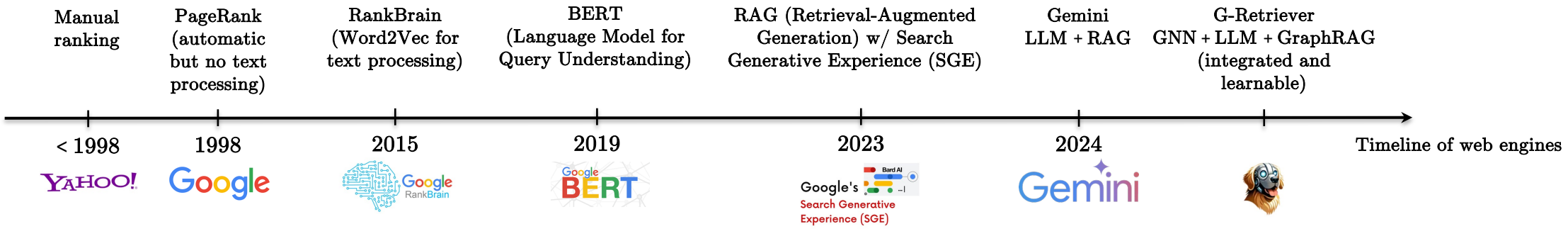

| | model | evidence | test_exact_pair | test_class_overlap | trainable_parameters | training_time_seconds |
|---|---|---|---|---|---|---|
| 0 | Full-GCN | BoW + Title/Abs + Anchor nodes | 0.6193 | 0.7919 | 154,119 | 2.13 |
| 1 | G-Retriever | Structural Feat + PCST + Prompt | 0.7005 | 0.8503 | 189,173 | 15.62 |

# Outline

- LLMs and GenAI
- Do LLMs make GNNs obsolete ?
- Combining LLMs and GNNs
- Using LLM for GNN reasoning
- Using GNN for LLM reasoning
- **Conclusion**

# Conclusion

- Textualized graphs make node and relation evidence readable to LLMs.
- G-Retriever combines query-conditioned retrieval, PCST, GNN and LLM answer generation.
- Applications include graph-grounded search, knowledge QA and conversational interfaces to structured graph data.
- Xiaoxin He's blogpost[1]: Practical introduction to graph retrieval-augmented generation.
- PyG implementation[2]: torch_geometric.nn.models.GRetriever.
- Nivida[3]: Advance AI driven drug discovery, delivers 5-10x better performance than RAG.




[1] https://medium.com/@xxhe/graph-retrieval-augmented-generation-rag-beb19dc30424
[2] https://pytorch-geometric.readthedocs.io/en/2.6.0/generated/torch_geometric.nn.models.GRetriever.html
[3] https://tv.qiagenbioinformatics.com/video/127772718/qiagen-and-nvidia-collaboration-to

Questions?