Optimizing Large Language Models for Academic Support: A Framework for Knowledge Graph Embedding and Cheat Sheet Generation

Yanjun He, Jiaye Liu, Xinrui Fu

Introduction

As AI tools like ChatGPT and Gemini become integral in education, their linear, isolated presentation of information limits deep comprehension in complex subjects. Students often struggle to connect related concepts in areas like math, science, and history, leading to cognitive overload, while educators invest extra time supplementing explanations. Research by Tan et al. (2024) and Liu et al. (2024) suggests that integrating context-rich, domain-specific knowledge graphs (KGs) with large language models (LLMs) could address these issues by offering structured, interrelated information.

Our research explores whether embedding model-generated KGs into LLMs improves educational efficiency by reducing cognitive load and enhancing comprehension. Using initial baseline evaluations, we will analyze the impact of KGs on the models' explanatory depth and accuracy. To measure effectiveness, varied true/false and short-answer questions will gauge comprehension improvement. Following Dai et al. (2020) and Tan et al. (2024), we will apply optimized embedding techniques to improve LLM interpretability. Ultimately, our project will develop a website generating personalized cheat sheets, providing students and educators with tailored, context-rich insights.

Background

Large Language Models (LLMs)

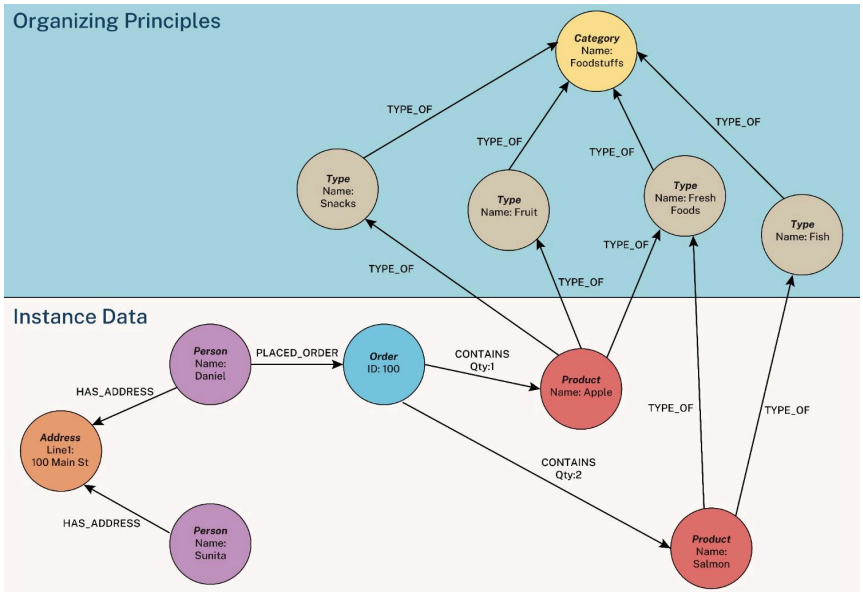
Large Language Models are AI systems that can communicate with humans. It is based on the Transformer architecture. It starts with processing text sequences by breaking them down into smaller parts through tokenization, then passes through an embedding layer that converts these tokens into vectors, which is a unique identity and also captures the semantic meaning. Positional

encoding is then applied to help the model recognize the position of tokens. The main part of the Transformer is a stack of encoder and decoder blocks, each repeated N times. Each encoder and decoder block includes a multi-head self-attention mechanism and a feed-forward neural network. The self-attention mechanism assigns weights to each word to show the importance and relevance, and multi-head means it is done multiple times in parallel. The feed-forward network then processes the output of the attention mechanism to refine. LLMs can be categorized into three types based on their architecture, encoder-only, decoder-only, and encoder-decoder models.

Training on this stack of encoders and decoders, LLMs are able to output the most ideal answer according to possibility. However, LLMs can also produce unsatisfactory outputs, often referred to as "hallucinations." These hallucinations can occur due to data limitations, inaccuracies, outdated information, or the probabilistic of language generation, where the model selects the most likely answer even if it's not factually correct.

Knowledge Graphs (KGs)

Knowledge Graphs stores factual information in structured data. It represents entities as nodes and the relationships between entities as edges. Entities can be anything meaningful nouns, like people, countries, subjects, etc. while edges show how these entities are related. It is often represented as triples (h, r, t), that head entity h, relationship r, and tail entity t. For example, ("UCI", "is located in", "Irvine") is a triple that represents the location of UCI in Irvine. Storing facts and information in structured data allows KG to be precise and interpretable. People can retrieve factual information and relevant information. Combining KGs with LLMs can improve the accuracy of the generated text because it is grounded by the structured, factual data provided by KGs.



An example knowledge graph showing nodes as circles and relationships as arrows. The instance data and organizing principles are highlighted for display.

Methodology

1. **Subject Selection and KG Design**

   Identify core university subjects (e.g.Math 2B) and select key topics (e.g. derivatives). Create knowledge graphs (KGs) that map essential concepts and relationships within each course.

2. **Question Development**

   Generate evaluation questions based on course syllabus, slides, and textbooks, including multiple-choice questions with explanations for correct answers. True/false questions targeting key concepts.

3. **Baseline LLM Evaluation**

   Assess models such as GPT-4, Llama 3.1 and Gemini 1.5 in their unmodified states. Use ROC analysis and confusion matrix evaluation for true/false questions to measure classification accuracy. Apply a scoring rubric for explanation quality in multiple-choice answers.

4. **Knowledge Graph Generation**

Using tools like Neo4j, Ontotext GraphDB, Stardog and Onethread, construct diverse KGs from course materials, ensuring each graph has a unique structure and format.

5. **Embedding KGs into LLMs**

   Embed each KG into both LLMs, applying optimal integration techniques to enhance content comprehension and answer accuracy.

6. **Comparative Analysis**

   Compare LLM performance before and after KG embedding to determine improvements in accuracy and explanation quality for each KG-LLM pairing.

7. **Optimal KG-LLM Pair Selection**

   Identify the best-performing KG-LLM combination to serve as the backend for a cheat sheet generator website, aimed at delivering precise and relevant academic support.

Timeline

Time	Activity
Fall 2024	**Subject Selection and KG Design** - Identify core subjects (e.g., Calculus 1) and select key topics (e.g., derivatives, integrals). - Begin designing initial knowledge graphs (KGs) for each subject, focusing on essential concepts and relationships. **Question Development** - Gather course syllabi, lecture slides, and textbooks for question creation. - Draft multiple-choice and true/false questions, including explanations for correct answers. **Baseline LLM Evaluation** - Assess baseline performance of models (GPT-4, Llama 3.1, Gemini 1.5). - Begin collecting ROC and confusion matrix data for true/false questions; refine scoring rubric for explanations.
Winter2025	**Knowledge Graph Generation** - Use tools (Neo4j, Ontotext GraphDB, Stardog, and Onethread) to complete initial KGs from course content. - Ensure diverse graph structures/formats for comparative analysis. **Embedding KGs into LLMs** - Embed each KG into selected LLMs, experimenting with integration techniques for optimized accuracy and comprehension.

	Preliminary Analysis - Conduct preliminary testing on KG-embedded LLMs using prepared questions. - Start comparative analysis between baseline and KG-embedded LLMs.
Spring 2025	**Comparative Analysis and Optimization** - Complete in-depth comparative analysis of LLM performance with and without KG embedding. - Identify the best-performing KG-LLM combinations based on accuracy and explanation quality. **Final Selection and Website Development** - Select the optimal KG-LLM pair for the cheat sheet generator website backend. - Begin designing and developing the website to deliver personalized cheat sheets based on the best-performing KG-LLM model. **Documentation and Reporting** - Document findings, methodologies, and results. - Prepare a final report and presentation.

Budget

Proposed Budget	**Price**
GPT - 4 ($0.18/call)	$540 (100 calls/week)
Llama 3.1 405b ($0.008/call)	$24 (100 calls/week)
Gemini 1.5 Pro (40.028/call)	$84 (100 calls/week)
Neo4j AuraDB Professional ($65/mo)	$195 (3 mo)
Copying/printing	$100
Total	**$943**

Reference

Dai, Yuanfei, et al. “A Survey on Knowledge Graph Embedding: Approaches, Applications and Benchmarks.” *Electronics*, vol. 9, no. 5, 2 May 2020, p. 750, https://doi.org/10.3390/electronics9050750.

Enzo. “What Is a Knowledge Graph?” *Graph Database & Analytics*, 18 July 2023, neo4j.com/blog/what-is-knowledge-graph/.

Liu, Haochen, et al. *Knowledge Graph-Enhanced Large Language Models via Path Selection*. 1 Jan. 2024, pp. 6311–6321, aclanthology.org/2024.findings-acl.376/, https://doi.org/10.18653/v1/2024.findings-acl.376. Accessed 5 Nov. 2024.

Tan, Xingyu, et al. “Paths-Over-Graph: Knowledge Graph Empowered Large Language Model Reasoning.” *ArXiv.org*, 2024, arxiv.org/abs/2410.14211. Accessed 5 Nov. 2024.