2025 ASEE Northeast Section Conference, March 22, 2025, University of Bridgeport, Bridgeport, CT, USA.

Optimizing an advanced contextual Question Answering model through the fine-tuning of a Pretrained Large Language Model.

SAFEENA KHANAM

Dept. of computer science

University of Bridgeport

Bridgeport, USA

skhanam@my.bridgeport.edu

***Abstract*— Pretrained Large Language Models (LLMs) are models with natural language understanding capabilities, but their generality can limit performance in specialized domains considering vastness and variance used for their training. In this project, we fine-tune a pre-trained LLM to adapt it for a specific context, improving its accuracy and relevance for context-specific queries. The fine-tuning process involves adding bias to the model parameters towards context and enabling it to answer questions and answers, so when a user questions anything to model it can prioritize the context over general data. By enhancing the model's ability to handle these types of specialized inquiries, this work demonstrates the effectiveness of fine-tuning in improving LLM performance for targeted applications.**

***Keywords—Large Language model; Fine Tuning; Pretrained models, optimization***

1. Introduction

 One of the key constraints of Large Language Models (LLMs) is that their vast training datasets, while extensive. The MIT Sloan Management Review highlights that if an LLM's training data lacks depth in a particular domain, it may struggle to generate relevant responses (1). Similarly, a study from the MIT Media Lab emphasizes that the opacity of training datasets can affect an AI model (2). A review in Cognitive Computation discusses how the effectiveness of LLMs is directly tied to the quality and scope of their training data, which can lead to limitations when addressing domain-specific queries (3). These studies collectively underscore that LLMs excel in general language tasks and their performance may falter when tasked with specialized.

1. EASE OF USE
2. *Milestones in Large Language Models*

 Large language models (LLMs) have demonstrated remarkable advancements in natural language understanding and generation. These models, trained on massive corpora, have enabled applications in machine translation, text summarization, and question-answering. However, most state-of-the-art LLMs, such as Open Ai’s GPT-4, remain proprietary, limiting accessibility for research and commercial use. To address this challenge,

open-source LLMs have emerged, offering transparency and reproducibility in model development. Among these, Databricks Dolly-v2-3B have gained significant attention for their efficiency and accessibility.

1. *Databricks Dolly-v2-3B*

 Dolly-v2-3B is an open-source, instruction-tuned LLM developed by Databricks for enterprise applications. Unlike proprietary models, it is freely available for commercial and research use. With 3 billion parameters, it is fine-tuned on instruction-based datasets to handle tasks like summarization, classification, and question-answering. Its key advantage is efficiency and accessibility, requiring fewer computational resources than larger models. This makes it suitable for deployment on consumer-grade GPUs.

1. *Facebook/bart-large-cnn*

 The facebook/bart-large-cnn model is a pre-trained BART variant specifically optimized for news summarization. It is designed to generate concise and coherent summaries while retaining key information from the input text. And This model is widely used for applications requiring automatic text summarization.

1. METHODOLOGICAL FRAMEWORK
2. *Multi-LLM approach*

 We are using the multi-LLM approach to switch between Dolly-v2-3B & Bloom-3B. From their key features we can understand Dolly is good for small queries, therefore we switch to Bloom for larger query size.

1. *System Functions:*
2. *Knowledge Graph functions:*
* *add\_knowledge:*

Stores new knowledge in the graph with semantic relationships.

* *get\_related\_knowledge*:

 Retrieves related data using breadth-first search within the knowledge graph.

1. *Memory Management Functions:*
* *add\_context:*

Encodes and stores query response pairs with embeddings.

* *retrieve\_context*:

Computes similarly between

 new queries andstored interactions to retrieve relevant

 context.

1. *Processing Functions:*
* *process\_query*:

 Implements the optimized flow with context, knowledge graph, and prompt optimization*.*

* + *process\_query\_regular*:

 Implements baseline processing without optimizations.

1. *Optimization Functions:*
	* *compress\_context:*

 Uses BART for summarization to reduce context length while preserving meaning

* + *extract\_keywords:*

Identifies key entities and terms from queries for knowledge graph retrieval.

1. *LLM Integration:*
* *generate\_response*(query):

Manages model selection and response generation based on query complexity.

1. *Implementation Parameters:*

 As this study focuses on contextual query optimization rather than model fine-tuning, we utilized pre-trained models with their default configurations:

* 1. *Memory Parameters:*
* *Context window size:* 10 entries maximum
* *Embedding dimension*: 384
	1. *Knowledge Graph Parameters:*
		+ *Graph structure:* Undirected NetworkX graph
		+ *Node attributes:* topic and description
		+ *Edge attributes:* related\_topics relationships
	2. *Summarization Parameters:*
* *Maximum summary length:* 50 tokens
* Minimum summary length: 10 tokens
* Sampling\_strategy: Deterministic (do\_sample=False)
	1. *LLM Generation Parameters*:
* Maximum new tokens: 100
* Sampling enabled: True (do\_sample=True)
1. *Knowledge Graph*

 A knowledge graph (KG) is employed to structure and retrieve contextual knowledge dynamically, enhancing large language models (LLMs) in domain-specific queries [8]. Unlike static embeddings, the KG enables incremental updates and efficient retrieval through structured relationships, modeled using Network. The graph G= (V, E) G = (V, E) G= (V, E) consists of nodes representing concepts and edges denoting semantic relationships.

 The add\_knowledge mechanism ensures new concepts are integrated coherently by verifying their existence before adding them as nodes.

 For context retrieval, the get\_related\_knowledge function utilizes breadth-first search (BFS) to extract conceptually linked nodes within a specified depth, ensuring relevant and explainable context retrieval [10]. The retrieved information is structured as (concept, relationship, related entity) (\text{concept}, \text{relationship}, \text {related entity}) (concept, relationship, related entity) tuples, improving factual consistency in LLM outputs.

 Performance is evaluated by measuring the efficiency of knowledge insertion and retrieval. Computational cost analysis of node additions and edge creation assesses scalability, while retrieval latency benchmarks demonstrate the advantage of structured context over traditional lookups [11]. The results validate this KG-based approach as an adaptive and efficient enhancement for LLMs.

1. *Memory Coefficient*

 A memory-efficient context retrieval mechanism is implemented to enhance large language models (LLMs) by dynamically managing and retrieving past interactions. This approach optimizes context retention while minimizing computational overhead, ensuring relevant historical information is efficiently utilized without excessive memory consumption [12].

The add\_context function encodes user input using the all-MiniLM-L6-v2 model from Sentence Transformers, storing the query-response pair along with its corresponding embedding. To maintain efficiency, the memory buffer is capped at ten entries, ensuring recent context is prioritized while removing older, less relevant data. This compact storage strategy prevents excessive resource utilization while preserving critical context

 For retrieval, the retrieve context function encodes the new query and computes cosine similarity scores against stored embeddings. The highest-scoring match is selected as the most relevant past interaction, allowing for efficient context-based response generation. This retrieval mechanism ensures that responses remain contextually relevant without requiring a full history scan, reducing computational complexity compared to some exhaustive search methods [14].

1. *Prompt Optimization*

 The Prompt Optimizer utilizes a transformer-based summarization model to efficiently compress textual context while preserving essential information. The methodology is implemented using the Hugging Face Transformers library, specifically leveraging the BART (Bidirectional and Auto-Regressive Transformer) model, pre-trained by Facebook [16].

* 1. *Transformer-Based Summarization:*

 The system employs a pre-trained BART model for abstractive summarization, enabling efficient text while preserving meaning. BART is a sequence-to-sequence model specifically designed for text generation tasks, including summarization and translation [17]. The model is initialized using the pipeline function from the Transformers library, with facebook/bart-large-cnn as the selected model, ensuring state-of-the-art performance.

 This approach guarantees high-quality text compression, maintaining semantic coherence and improving the efficiency of downstream LLM-based tasks. By leveraging BART’s transformer-based architecture, the system effectively reduces input size, making it particularly useful for applications with token limitations or memory constraints. Furthermore, the deterministic nature of this summarization process ensures consistent and reliable outputs.

1. ARCHITECTURE OF PROJECT

 The below Fig. 1. Architecture and data flow are key elements of the project. The sole goal is to maintain optimized flow and feed relevant context to the pre-trained model from the knowledge graph and weigh in its relevance using memory coefficient.



 Fig. 1. Architecture of Code

The above Fig. 1 illustrates a query optimization queries before feeding them into a Large Language Model (LLM). The key components of this system include:

*A. Query Input:*

The system receives a user query, which is processed

using additional contextual and memory-based information to improve its structure and clarity.

*B. Context and Knowledge Graph:*

 Context extraction allows the system to identify relevant background information that can improve response generation. From the Fig 1. Knowledge Graph is employed to enrich the query by structuring data, linking related concepts, and retrieving essential information from a predefined knowledge base.

* 1. *Memory Coefficient:*

 This component stores historical interactions, improving response consistency by incorporating past data into the query optimization process.

1. *Prompt Optimizer:*

 The Knowledge Graph and Memory Coefficient outputs are combined and optimizing the prompt before it is sent to the LLM.

1. *LLM Processing and Response:*

 From the optimized prompt is processed by the LLM, generating a refined response based on the enriched query.

1. RESULTS

 The results are captured by comparing responses between the optimized approach, which provides clear and detailed answers, and the regular flow, where queries are directly fed to the LLM.

TABLE 1. TEST QUERIES



 The above TABLE 1. compares optimized and regular responses regarding Safeena Khanam and details like her current position, research papers, specializers and University details etc. By refining the responses, the optimized approach enhances offer clear, detailed, and informative, clarity, precision, and engagement, making the information more useful for the reader. Meanwhile, the regular responses remain vague and unstructured, limiting their effectiveness. This comparison highlights the importance of structured and context-aware answers in delivering accurate and valuable information.

 TABLE 2. REGULAR AND OPTIMIZED QUERY



 The above TABLE 2. represents a comparison between Regular and Optimized query processing, focusing on time taken and relevance while considering the impact of context availability.

Here’s an analysis of the data:

* 1. *Query & Context Provided:*
		+ Some queries are processed with context (true),

while others are processed without context (false).

* 1. *Time Taken (Regular vs. Optimized):*
* Regular processing times range between 22.1s

and 32.965s when context is provided and 23.49s to 28.97s when no context is used.

* Optimized processing tends to take longer, ranging from 34.22s to 68.854s, indicating that optimization increases processing overhead.
	1. *Relevance (Regular vs. Optimized):*
* The Regular method produced relevant results for only one query (Query 1), indicating limited effectiveness.
* The Optimized method improved relevance in two additional cases (Queries 2 and 3), highlighting its ability to refine responses.
* However, both methods failed to generate relevant responses for Queries 4 and 5, suggesting that certain queries may require alternative optimization strategies or more comprehensive training data.
	1. *system performance was evaluated using the following metrics:*
* Measured in seconds from query submission to response generation.
* For queries about Safeena Khanam, responses were compared against the ground truth in the knowledge graph.
* Evaluated through follow-up questions that referenced previous interactions
1. CONCLUSION

 In the following sample outputs, we could see our optimized flow was successful in capturing accurate results and relatively quite faster than regular usage of pre-trained LLM. With better validation metrics and large-scale Curated query list, this could be proven in a more comprehensive manner. Further testing with diverse datasets and real-world scenarios will help solidify these findings. Additionally, fine-tuning the model with domain-specific data and optimizing inference strategies could further enhance accuracy and efficiency, making it a viable solution for large-scale deployments.

REFERENCES

[1] MIT Sloan Management Review. (2023). The Working Limitations of Large Language Models. Retrieved from sloanreview.mit.edu

[2] MIT Media Lab. (2023). Transparency is Often Lacking in Datasets Used to Train Large Language Models. Retrieved from media.mit.edu

[3] Springer. (2024). Challenges in Large Language Model Training and Specialization. Cognitive Computation. Retrieved from link.springer.com

[4] Databricks, “Dolly: The First Open Instruction-Tuned LLM,” Apr. 2023. [Online]. Available: https://huggingface.co/databricks/dolly-v2-3b

[5] Databricks, “Fine-tuning Dolly-v2 for Task-Specific Applications, 2023. [Online]Available:

https://www.databricks.com/blog/2023/04/12/dolly-first-open-instruction-tuned-llm.html

[6] BigScience, “BLOOM: A 176B Parameter Open-Access Multilingual Language Model,” Nov. 2022. [Online]. Available: https://huggingface.co/bigscience/bloom-3b

[7] T. Wolf et al., “BigScience BLOOM: Open-Source Large Multilingual Language Models,” arXiv:2211.05100, 2022. [Online]. Available: https://arxiv.org/abs/2211.05100

[8] A. Bordes et al., “Translating embeddings for modeling multi-relational data,” NeurIPS, 2013.

[9] C. Zhang et al., “Deep learning-based knowledge graph embedding: A survey,” IEEE TKDE, 2022.

[10] M. Nickel et al., “A review of relational machine learning for knowledge graphs,” Proc. IEEE, 2016.

[11] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” KDD, 2016.

[12] J. Devlin et al., “BERT: Pre-training of deep bidirectional transformers for language understanding,” NAACL-HLT, 2019.

[13] S. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using Siamese BERT-networks,” EMNLP, 2019.

[14] Q. Zhu et al., “Efficient context retrieval in dialogue systems using neural embedding similarity,” ACL, 2021.

[15] H. Jeon et al., “Memory-efficient transformer architectures for long-context NLP,” ICLR, 2022.

[16] Y. Liu et al., "Robustly Optimized BART: A Large-Scale Transformer for Text Generation," in Proceedings of the Association for Computational Linguistics (ACL), 2020.

[17] M. Lewis et al., "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," in Proceedings of the Annual Meeting of the Association.