Beyin: An Integrated Framework for Multi-Expert LLM Orchestration with RL-Based Routing

Prof. Kavita Babalad, Janhavi Sachdev, Chaithali S.K
REVA University

***Abstract*— Project Beyin presents a novel framework that integrates multiple domain-specific Large Language Models (LLMs) with an intelligent routing mechanism. This paper introduces a reinforcement learning-based dynamic query routing system designed to address subject-specific queries with high precision. We implement Parameter-Efficient Fine-Tuning (PEFT) techniques, specifically Low-Rank Adaptation (LoRA), alongside sophisticated retrieval mechanisms to deliver factually grounded, context-aware responses. Our methodology encompasses three key components: (1) fine-tuning multiple LLMs for targeted domains using specialized datasets, (2) implementing an advanced Retrieval-Augmented Generation (RAG) pipeline with embedding-based retrieval and contextual re-ranking, and (3) developing a Proximal Policy Optimization (PPO) based routing system that dynamically directs queries to the appropriate domain expert model. Experimental results demonstrate that Beyin effectively reduces hallucinations, enhances contextual relevance, and scales efficiently across diverse applications including healthcare, education, and enterprise knowledge management. This research addresses critical gaps in existing frameworks by providing an integrated solution for domain-specific AI challenges. Future work will focus on incorporating real-time feedback mechanisms and expanding multimodal integration capabilities.**

***Index Terms*— Retrieval Augmented Generation (RAG), AI Agents, Large Language Models (LLM), Knowledge Graphs (KG)**

I. INTRODUCTION

Advancements in artificial intelligence (AI) have significantly transformed natural language processing (NLP), with Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) pipelines at the forefront. These technologies form the backbone of various applications, including conversational agents, automated content generation, and domain- specific knowledge systems, offering remarkable capabilities in understanding and generating human- like text.

A. Background on Large Language Models (LLMs)

LLMs are powerful deep learning models designed to process and generate human language. Built on transformer architectures, these models leverage attention mechanisms to capture the semantic and syntactic intricacies of language, enabling tasks like summarization, question answering, and contextual text generation. Their development has evolved from early rule-based systems to neural network-based transformers, with landmark models such as BERT, GPT, and T5 shaping the field. By training on extensive datasets comprising billions or even trillions of tokens, LLMs have achieved state-of-the-art performance across diverse NLP tasks.

Despite their success, LLMs face inherent limitations. Their reliance on static training data restricts their ability to provide up-to-date information or domain-specific knowledge. Moreover, their large size and computational demands pose challenges for efficient deployment in real-world applications.

B. Background on Retrieval-Augmented Generation (RAG)

RAG pipelines address these limitations by combining retrieval-based techniques with generative modeling. In a typical RAG framework, relevant information is retrieved from external knowledge bases, such as databases or document repositories, and provided as context to an LLM for response generation. This integration enhances factual accuracy and relevance while maintaining the fluency of generative models. RAG pipelines are particularly beneficial in

domains requiring up-to-date or highly specialized information, such as medicine, legal research, and technical support.

C. Motivation for Topic-Oriented Approaches

While LLMs and RAG pipelines offer substantial potential, a key challenge lies in tailoring these systems to address specific topics effectively. Fine- tuning multiple LLMs for domain-specific tasks, coupled with intelligent query routing, provides an opportunity to optimize performance and resource utilization. The use of AI agents to dynamically analyze user queries and select the most appropriate model introduces an additional layer of adaptability and precision, ensuring responses are contextually relevant and domain-appropriate.

D. Research Gap

Existing studies on RAG pipelines and LLMs have primarily focused on single-model implementations, with limited exploration of integrating multiple fine- tuned models within a unified system. Additionally, there is a lack of frameworks for AI-driven query routing, which could leverage model diversity to enhance the performance across specialized domains.

II. LITERATURE SURVEY

A. Fine-Tuning Methods for Large Language Models (LLMs)

Fine-Tuning Methods for Large Language Models (LLMs) refers to specialized techniques used to adapt pre-trained language models to specific tasks, do- mains, or datasets. These methods involve updating the model's weights or utilizing additional parameters based on a smaller, task-specific dataset to improve performance on a given objective while retaining the knowledge learned during pre-training. Fine-tuning can enhance model accuracy, efficiency, and applicability for tasks such as text classification, question answering, or language translation. Popular fine-tuning strategies include full fine-tuning, prompt tuning, LoRA (Low-Rank Adaptation), and prefix tuning.

Fine-tuning has become an essential strategy for adapting large pre-trained language models to specific domains or tasks. Existing approaches range from traditional fine-tuning techniques to more advanced methods designed to address challenges such as computational inefficiency and overfitting.

Fine-tuning pre-trained LLMs is crucial for adapting them to specific tasks or domains, enhancing their performance and applicability. Several approaches have been developed, each with its strengths and weaknesses. Table 1 provides a com- parison of these fine-tuning methods.

1) Instruction Fine-Tuning

Instruction fine-tuning involves training models using datasets that provide explicit task instructions, improving the model's ability to interpret and adhere to human-provided prompts. This technique has shown success in tasks such as summarization and question answering. [1][33]

2) Parameter-Efficient Fine-Tuning (PEFT) PEFT methods, including Low-Rank Adaptation (LoRA) and Prompt Tuning, aim to reduce the computational burden of full model fine-tuning. By updating only a subset of the model's parameters, these approaches allow for efficient adaptation without requiring extensive computational resources. [2]
3) Task-Specific and Domain-Specific Fine-Tuning Tailoring models to excel in particular tasks or domains involves using curated datasets relevant to the specific area of interest. This approach enhances the model's performance on specialized tasks but may lead to challenges such as overfitting or catastrophic forgetting, where the model loses its general-purpose capabilities. [3]
4) Behavioral and Sequential Fine-Tuning Behavioral fine-tuning incorporates real-world user interaction data to improve conversational capabilities. Sequential fine-tuning adapts models incrementally across related tasks, mitigating risks of catastrophic forgetting while maintaining generalizability. [4]

B. Retrieval-Augmented Generation (RAG) Frame- works

RAG frameworks enhance the capabilities of LLMs by integrating retrieval mechanisms that fetch relevant information from external databases, thereby grounding the generated responses in factual and up-to-date knowledge. This approach addresses challenges such as hallucination and outdated information inherent in standalone LLMs. [5]

The RAG process typically involves two stages: retrieval and generation. In the retrieval stage, relevant documents or knowledge snippets are fetched based on the user's query. In the generation stage, the retrieved information guides the LLM to produce coherent and contextually accurate responses. This framework has been applied in various domains, including question answering systems and personalized education, by dynamically incorporating external knowledge to enhance response quality. [6]

C. Dynamic Query Routing

Dynamic query routing plays a critical role in the effectiveness of multi-model AI systems by ensuring that queries are directed to the most appropriate domain-specific models. Traditional approaches primarily relied on rule-based heuristics or domain classifiers. These methods, while interpretable and straightforward, were limited by their rigidity and in- ability to adapt to unseen or ambiguous query patterns. To address these limitations, researchers have increasingly turned to reinforcement learning (RL), which allows systems to learn optimal routing strategies from interaction data. In such systems, query routing is typically modeled as a Markov Decision Process (MDP), where the state is derived from semantic embeddings of the input query, the action represents a choice among available models, and the reward is
based on the correctness of the routing decision.
Early work in this field utilized Deep Q-Networks (DQN) to map query features to routing decisions.

TABLE I: COMPARISON OF FINE-TUNING METHODS FOR LLMs

Feature	Instruction Fine-Tuning	Parameter-Efficient Fine-Tuning (PEFT)	Task-Specific/Domain-Specific Fine-Tuning	Behavioral and Sequential Fine-Tuning
Description	Training models using datasets with explicit task instructions to improve adherence to human prompts.	Reducing the computational burden of full fine-tuning by updating only a subset of the model's parameters. Examples: LoRA, Prompt Tuning	Tailoring models to excel in specific tasks or domains using curated datasets.	Behavioral fine-tuning incorporates real-world user interaction data to improve conversational capabilities. Sequential fine-tuning adapts models incrementally across related tasks, mitigating risks of catastrophic forgetting while maintaining generalizability.
Goal	Improve the model's ability to understand and follow instructions.[35]	Enable efficient adaptation of LLMs without ex-tensive computational re-sources.	Enhance the model's performance on specialized tasks within a specific do- main.	To improve conversational capabilities and reduce catastrophic forgetting respectively
Advantages	Success in tasks like sumarization and question answering.	Allows for efficient adaptation without requiringextensive	Enhances the model's performance on specialized tasks.	Improved conversational capabilities and prevention of catastrophic forgetting.

		computational resources.		
Disadvantages	Requires well-defined and comprehensive instruction datasets.	May not achieve the same level of accuracy as full fine-tuning in all cases.[35]	May lead to overfitting or catastrophic forgetting, where the model loses its general-purpose capabilities.	Requires real world user data to ensure relevance and accuracy.[33]
Example Applications	Training a chatbot to follow specific dialogue guidelines or a summarization model to adhere to a particular summarization style.	Fine-tuning a model for code generation on a low- resource device or adapting a language model for a specific writing style with limited data.	Creating a medical chat-bot that understands medical terminology or a legal document summarizer that is familiar with legal jargon.	Adapting customer ser-vice bots to handle com- plex user inquiries, or creating a series of educational modules.

While effective, these models often struggled with stability and required substantial tuning. More recently, policy gradient methods such as Proximal Policy Optimization (PPO) have emerged as superior alternatives due to their robustness and efficiency in policy learning. These models typically include a semantic embedding encoder—often based on transformer architectures like BERT or Sentence-BERT—to convert queries into meaningful state representations, followed by a policy network that outputs the probabilities for each routing option.

Reward design in RL-based routers typically starts with a simple binary reward (+1 for correct routing, -1 otherwise), though more advanced systems integrate soft reward mechanisms based on user satisfaction scores or model performance metrics. This flexibility allows RL agents to adapt to complex, evolving rout- ing criteria over time, making them especially suitable for dynamic environments like retrieval-augmented generation (RAG) frameworks, conversational assistants, and intelligent tutoring systems.

Despite the advantages, RL-based routing systems face several challenges, including sample inefficiency, reward sparsity, and the computational overhead of training in high-dimensional spaces. Nevertheless, on- going advancements in model architecture, reward engineering, and continual learning strategies continue to improve the practicality and performance of these systems.

D. Learning and Adaptation

Through machine learning techniques such as reinforcement learning, AI agents refine their performance over time. They utilize feedback from users and system interactions to adjust their strategies, maintaining relevance and accuracy in dynamic environments. [8] Integrating AI agents with fine-tuned LLMs and RAG frameworks offers a robust solution for developing personalized, context-aware AI systems. This integration enhances the system's ability to manage domain-specific inquiries effectively, providing users with accurate and relevant information tailored to their specific needs.

III. METHADOLOGY

A. Overview of Fine-Tuning Multiple Large Language Models (LLMs)

Fine-tuning multiple large language models (LLMs) plays a crucial role in this research by enabling the adaptation of pre-trained, general-purpose models to specific domains. This process involved selecting several state-of-the-art models, applying various fine- tuning techniques, and integrating the models into a Retrieval-Augmented Generation (RAG) pipeline to achieve optimized performance on topic-specific tasks. The chosen models include GPT-4 for its broad NLP capabilities, LLaMA for efficient language understanding, Mistral for handling complex linguistic tasks, FLAN-T5 for instruction-following tasks, and

Cohere Command, which excels in semantic understanding for retrieval-based systems. These models were fine-tuned on domain-specific datasets tailored to fields such as biology, healthcare, and technical support.[31]

Fine-tuning techniques included full fine-tuning, which adjusts all parameters for maximum accuracy; parameter-efficient fine-tuning (PEFT) approaches such as LoRA and adapter-based tuning, which reduce computational costs by updating fewer parameters; instruction tuning, which improves adapt- ability to tasks involving explicit instructions; and supervised fine-tuning (SFT) using labeled domain- specific datasets. To ensure task relevance, custom datasets were curated and preprocessed for each do- main. The biology dataset was compiled from aca- demic journals, textbooks, and FAQs, the healthcare dataset from medical research and clinical guidelines, and the technical dataset from troubleshooting guides and manuals. The preprocessing steps included noise removal, tokenization, and formatting of the data for fine-tuning frameworks ([1], [5]).

B. Role and Architecture of AI Agents in Topic-Oriented Approaches:

1) Role of Dynamic Query Routing:
In multi-model AI systems like Beyin, where multiple fine-tuned models specialize in distinct domains such as healthcare, education, and general knowledge, effective query routing is essential to ensure the relevance, accuracy, and efficiency of generated responses. Traditional static routing methods based on keyword matching or rule-based logic often fall short when handling complex or ambiguous queries that span multiple domains.
Dynamic Query Routing, empowered by Reinforcement Learning (RL), offers a data-driven, adaptive solution. Instead of relying on predefined rules, the system learns to route queries through trial and feedback. By modeling the routing process as a sequential decision-making problem, the RL-based router can adapt to do- main shifts, improve over time, and generalize to unseen query types. This routing strategy ensures that each user query is directed to the model most capable of producing contextually appropriate and high-quality responses.
2) Architecture of the RL-Based Routing System
The architecture for dynamic query routing using RL consists of the following key components:

a) Query Embedding Module:
The system begins by semantically encoding the incoming query using Sentence-BERT (all-MiniLM-L6-v2), producing a dense vector representation. This embedding captures the query's intent, context, and domain-specific signals, forming the state input for the RL agent.

b) Custom Gymnasium Environment:
A simulated environment is built using the Gymnasium framework, where the agent interacts with the query routing task. The environment is defined by:

- State: Query embeddings representing the semantic state.
- Action Space: Discrete set of available models (e.g., Medical, Educational, General).
- Reward Function:
  – +1 for correct routing (i.e., selected model generates the most relevant answer)
  – [−1] for incorrect routing.

This setup allows the RL agent to learn optimal routing behavior via trial-and-error interaction

3) Feedback Loop and Continual Learning To enhance adaptability, the system incorporates a feedback loop wherein the outcome of each query resolution is logged. The feedback (e.g., user rating, model confidence, or correctness label) is used to retrain or fine-tune the RL policy periodically. This ensures the router remains robust against changing query distributions or evolving domain needs.
4) Policy Learning with Proximal Policy Optimization (PPO) The core learning mechanism is implemented using the Proximal Policy Optimization (PPO) algorithm. PPO is chosen for its stability and sample efficiency in continuous action environments. The policy network takes the query embedding as input and outputs a probability distribution over the action space (i.e., models). The agent selects the model with the highest expected return.

Mathematically, The policy network outputs probabilities for selecting each model:

$$\pi_\theta(a|s) = P(a \mid s;\theta)$$

The goal is to optimize the policy using the PPO objective, which incorporates a clipped surrogate loss to constrain large policy updates:

$$L^{CLIP}(\theta) = \hat{E}_t \left[\min \left[r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right] \right]$$

where $r_t(\theta)$ is the probability ratio between the new and old policy, $\hat{A}_t$ is the estimated advantage, and ϵ is a hyperparameter (commonly $0.1 - 0.2$).

C. *Retrieval-Augmented Generation (RAG) Pipeline*

The integration of fine-tuned large language models (LLMs) into a Retrieval Augmented Generation (RAG) framework is central to this research, allowing the generation of responses grounded in factual data through advanced retrieval mechanisms. Retrieval plays a crucial role in ensuring that the generated outputs are accurate and contextually relevant. Several

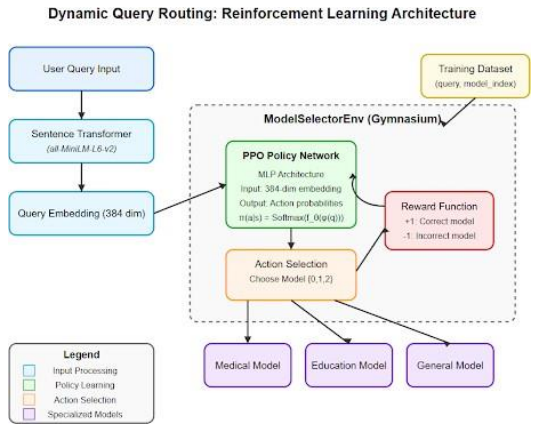

Fig. 1. Architecture of Dynamic Query routing

retrieval techniques are used including embedding-based retrieval, where semantic similarity searches identify relevant content by comparing query and document embeddings. Hybrid search, which combines keyword-based and embedding-based retrieval, enhances accuracy by leveraging both approaches. Additionally, contextual re-ranking is used to prioritize high-quality results by assessing their relevance within the given context ([5]).

Once the relevant content is retrieved, it is fed into fine-tuned LLMs alongside user queries to ensure that the generated outputs are factually grounded and con- textually appropriate. Fine-tuned models are specifically tailored to different domains, such as healthcare and technical support, to improve their performance in providing domain-specific insights. By deploying models fine-tuned in task-relevant datasets, the system ensures that responses are not only accurate, but also highly relevant to the specific needs of the user ([1]).

To ensure continuous improvement in the quality of the retrieval and response, a feedback loop is implemented. AI agents monitor user interactions and performance metrics, enabling iterative refinement of both retrieval mechanisms and model outputs. This feedback-driven approach allows the system to learn from user behavior and improve over time, ensuring that the RAG framework remains effective in delivering accurate and high-quality responses across various applications ([1], [5]).

D. *Evaluation Metrics*

The performance of the proposed system was evaluated using the following metrics:

Accuracy: Comparison of outputs with ground truth data. • Relevance: Contextual alignment of the content retrieved and generated.

User Satisfaction: Feedback from end users on the system's utility and responsiveness ([1], [5]).

IV. WORK DONE AND RESULTS ANALYSIS

A. *Fine Tuning using LoRA*

Fine-tuning TinyLlama (1.1B) or any transformer-based language model follows the principles of Gradient Descent, Backpropagation, and Loss Optimization.

1) Preliminaries: Transformer Model Basics TinyL- lama is a decoder-only Transformer model.

A Transformer processes token sequences X = (x, x, ..., x) and learns to predict the next token y given previous tokens.

2) Tokenization and Embedding Layer:

Each token x is mapped to an embedding vector $E(x)^{d}$:

$$h_0 = E(x_t) + P_t$$

where:

- $E(x_t)$ is the token embedding.
- P_t is the positional encoding.

3) Self-Attention Mechanism;

The multi-head attention mechanism learns token dependencies. It computes:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where:

- Q,K,V are the query, key, and value matrices
- W_Q, W_K, W_V are learnable weight matrices
- d_k is the dimension of the key vectors

The self-attention output is:

$$H_{\text{attn}} = \sum_{h=1}^{H} \text{Attention}_h(Q, K, V)W_O$$

where H is the number of attention heads.

4) Feedforward Network: Each token representation passes through a feedforward network (FFN):

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

where:

- W_1, W_2 and b_1, b_2 are learnable parameters.

5) Causal Language Modeling (CLM) Objective TinyL- lama is trained using Causal Language Modeling (CLM), which maximizes the likelihood of predicting the next token:

$$L = -\sum_{t=1}^{N} \log P(y_t | y_{<t}; \theta)$$

where:

- $P(y_t | y_{<t}; \theta)$ is the model's predicted probability for token y_t.
- θ represents model parameters.

Using Cross-Entropy Loss:

$$L = -\sum_{t=1}^{N}\sum_{v=1} y_{t,v} \log \hat{y}_{t,v}$$

where:

- $y_{t,v}$ is 1 if the ground truth token is v, else 0.
- $\hat{y}_{t,v}$ is the predicted probability.

6) LoRA (Low-Rank Adaptation) for Fine-Tuning: LoRA fine-tuning modifies only low-rank weight matrices in attention layers, reducing trainable parameters. Instead of modifying full weights W, LoRA updates:

$$\Delta W = AB$$

where:

- $A \in \mathbb{R}^{d \times r}$, $B \in \mathbb{R}^{r \times d}$.
- $r \ll d$ (low-rank adaptation).

The updated weights are:

$$W' = W + \Delta W$$

7) Backpropagation and Gradient Descent: During training, we compute gradients of the loss L w.r.t. model parameters:[30]

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta L$$

where:

- η is the learning rate,
- $\nabla_\theta L$ is the gradient of the loss.

Using AdamW Optimizer, we update weights with momentum and weight decay:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t$$

where:

- m_t, v_t are moving averages of gradients.

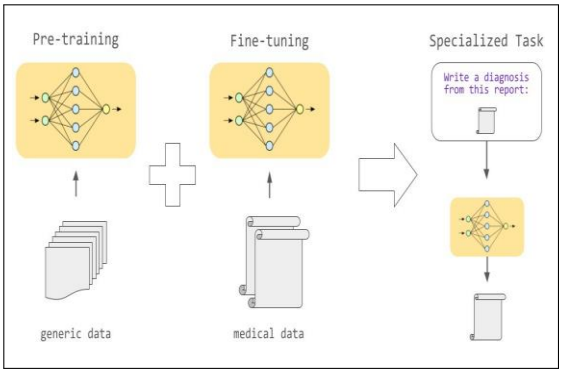

Fig. 2. Process of Fine-Tuning

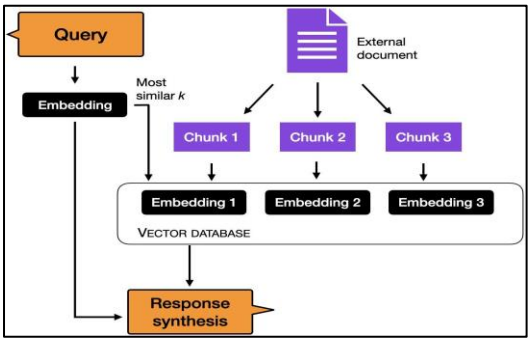

Fig. 3. Chunking in RAG

B. *Fine-Tuning using RAG*

The workflow was initiated with the acquisition and preprocessing of a domain-specific dataset, namely the BioASQ.org file, which comprises biomedical questions and their corresponding contextual snippets. Each question contains a list of textual snippets manually annotated as relevant to the query. These snippets formed the basis for our retrieval corpus.

To transform the raw text data into a format amenable to semantic similarity search, we employed the all-MiniLM-L6-v2 Sentence Transformer model [32], a pre-trained language model known for its efficiency and competitive

semantic representation capabilities. This model encoded each snippet into a 384- dimensional dense vector, thereby mapping textual data into a continuous vector space where semantically similar texts reside in close proximity. The resultant embeddings were converted to 32-bit floating point format (float32) to ensure compatibility with the FAISS indexing library used in subsequent stages.

For indexing and retrieval, we utilized FAISS (Face-book AI Similarity Search), an efficient similarity search library capable of handling large-scale vector data. Specifically, the IndexFlatL2 index was chosen

to facilitate retrieval based on the L2 (Euclidean) distance metric, which is commonly used in semantic search tasks. The embeddings of all snippets were added to the FAISS index, enabling fast approximate nearest-neighbor search.

Upon receiving a user query, our system encodes the query into its corresponding vector representation using the same Sentence Transformer model. This vector is then used to query the FAISS index to retrieve the top-k most semantically similar snippets from the indexed corpus. These snippets serve as a contextual foundation for generating the final answer, effectively augmenting the query with relevant knowledge. The final step involves the integration of an advanced generative language model to synthesize the answer. We utilized Claude API to access the llama3- 8b-8192 model, a variant of the LLaMA-3 architecture fine-tuned for open-domain generation. The retrieved snippets and the original query were composed into a structured prompt that instructed the model to generate answers based strictly on the provided context. A system-level prompt was also included to guide the model toward generating responses from the perspective of a medical expert.

1) Embedding Mapping Function

Given a sentence s S, the embedding function f is:

$$f : S \to \mathrm{R}^{d}$$

where $d = 384$ for `all-MiniLM-L6-v2`.

2) Similarity Function (Euclidean Distance)

For query vector $\vec{q} \in \mathrm{R}^{d}$ and snippet vector $\vec{s}_i \in \mathrm{R}^{d}$:

$$D(\vec{q}, \vec{s}_i) = \| \vec{q} - \vec{s}_i \|_2 = \sqrt{\sum_{j=1} (q_j - s_{ij})^2}$$

FAISS uses this to find nearest vectors.

3) RAG Prompt Composition

$$\mathrm{Prompt}(q, \{s_i\}_{i=1}^{k}) = \mathrm{Template}\left(\bigoplus_{i=1}^{k} s_i, q\right)$$

C. Dynamic Query Retrieval

Purpose and Functionality:

The retrieval module performs two primary functions:

a) Semantic embedding of queries and documents, transforming them into high-dimensional vectors;

b) Similarity search over a pre-indexed vector database to retrieve top-k relevant results (e.g., model descriptors, knowledge chunks, or training feedback samples).

2) Query Embedding We use a sentence transformer model, specifically all-MiniLM-L6-v2, to generate vector embeddings for all queries:

$$q = f(q), qR^{[d]}$$

where:

- f is the Sentence-BERT encoder,
- q is the input query,
- q is the resulting embedding vector,
- d=384 (embedding dimensionality for MiniLM).

3) Retrieval via Cosine Similarity:

Given a query vector q and a set of document/model/domain vectors d 1 ,d 2 ,. . . ,d n , we compute cosine similarity:[34]

$$\mathrm{sim}(\mathrm{q}, \mathrm{d}_i) = \frac{\mathrm{q} \cdot \mathrm{d}_i}{\|\mathrm{q}\| \cdot \|\mathrm{d}_i\|}$$

This metric returns a score in [–1, 1], where values closer to 1 indicate higher semantic alignment. The top-k most similar vectors are selected:

$$\mathrm{TopK}(\mathrm{q}) = \underset{i \in \{1,\ldots,n\}}{\mathrm{argmax}}\ \mathrm{sim}(\mathrm{q}, \mathrm{d}_i)$$

Hybrid Retrieval: In some configurations, a hybrid search may be employed combining cosine similarity with BM25 (lexical matching):

$$\mathrm{Score}(q, d) = \alpha \cdot \mathrm{BM25}(q, d) + (1 - \alpha) \cdot \mathrm{sim}(q, d)$$

where α ∈[0, 1] is a tunable hyperparameter. This approach balances semantic matching with token-level relevance.

5) Re-ranking and Relevance Filtering After retrieval, a lightweight re-ranking model or rule-based filter can re-score the top-k results using domain-specific context or attention models:

$$\mathrm{ReRank}(d_i) = \mathrm{ContextMatch}(q, d_i) + \mathrm{ConfidenceScore}(d_i)$$

This ensures robustness in ambiguous or borderline queries and allows dynamic relevance enhancement.

TABLE I: COMPARISON OF DIFFERENT METHODS

Method	Routing Accuracy	Training Overhead	Inference Latency

Rule-based	78%	N/A	0.5ms
Embedding Similarity	83%	Medium	1.2ms
Decision Tree	85%	Low	0.8ms
RL-based (Ours)	92%	High	1.0ms

V. CONCLUSIONS

The landscape of fine-tuning Large Language Models, RAG frameworks, and AI agent systems reflects the ongoing evolution of natural language processing technologies. While fine-tuning enables domain adaptation, RAG frameworks enhance relevance by integrating external knowledge. AI agents further amplify these capabilities by orchestrating dynamic and context-aware query routing, ensuring that the most appropriate tools are used for each task. Together, these innovations are reshaping the way AI systems tackle complex topic-specific challenges.

REFERENCES

[1] Wei, J., et al. (2022). "Finetuned Language Models Are Zero-Shot Learners." Retrieved from https://arxiv.org/abs/2109.01652

[2] Hu, E., et al. (2021). "LoRA: Low-Rank Adaptation of Large Language Models." Retrieved from https://arxiv.org/abs/2106.09685

[3] Lee, J., et al. (2020). "BioBERT: a pre- trained biomedical language representation model for biomedical text mining." Retrieved from https://academic.oup.com/bioinformatics/article/36/4/1234/5566506

[4] Zoph, B., et al. (2022). "ST-MoE: Designing Stable and Transferable Sparse Expert Models." Retrieved from https://arxiv.org/abs/2202.08906

[5] Lewis, P., et al. (2020). "Retrieval-Augmented Genera- tion for Knowledge-Intensive NLP Tasks." Retrieved from https://arxiv.org/abs/2005.11401

[6] IBM Research. (n.d.). "Retrieval-Augmented Generation: Leveraging the Power of Retrieval and Generation in Language Models." Retrieved from https://research.ibm.com/blog/retrieval-augmented-generation- RAG

[7] Wang, X., et al. (2021). "Dynamic Query Routing in Multi-Domain Task-Oriented Dialog." Retrieved from https://arxiv.org/abs/2104.08419

[8] Silver, D., et al. (2017). "Mastering the game of Go without human knowledge." Retrieved from https://www.nature.com/articles/nature24270

[9] Rastogi, A., Zang, X., Sunkara, S., Gupta, R., Khai- tan, P. (2019). Towards Scalable Multi-Domain Conver- sational Agents: The Schema-Guided Dialogue Dataset. arXiv:1909.05855.

[10] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W. (2016). OpenAI Gym. arXiv:1606.01540.

[11] Zhang, H., Liu, Y., Wu, Z. (2022). Adaptive Model Selection Using Reinforcement Learning in Intelligent Tutoring Sys- tems. Proceedings of ACL 2022.

[12] Reimers, N., Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv:1908.10084.

[13] Chen, M., Liu, Z., Xu, Y. (2020). Learning Query Routing with User Feedback for Federated Search. IEEE Transactions on Knowledge and Data Engineering.

[14] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv:1707.06347.

[15] Nentidis, A., et al. (2023). Results of the eleventh edition of the BioASQ Challenge. In Proceedings of the BioASQ Workshop.

[16] Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS 2020. , N., Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. EMNLP 2019.

[17] Johnson, J., et al. (2019). Billion-scale similarity search with GPUs. IEEE Transactions on Big Data.

[18] Jin, Q., et al. (2019). PubMedQA: A Dataset for Biomedical Research Question Answering. EMNLP 2019.

[19] Touvron, H., et al. (2023). Llama 3: Our Best Open Model. Technical report, Meta AI.

[20] Gururangan, S., et al. (2020). Don't Stop Pre Training: Adapt Language Models to Domains and Tasks. ACL 2020.

[21] Lee, J., et al. (2020). BioBERT: a pre-trained

biomedical language representation model for biomedical text mining. Bioinformatics.
[22] Karpukhin, V., et al. (2020). Dense Passage Retrieval for Open-Domain Question Answering. EMNLP 2020.
[23] Chen, D., et al. (2021). Evaluating Large Language Models Trained on Code. ArXiv, abs/2107.03374.
[24] Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks Lewis et al., 2020 https://arxiv.org/abs/2005.11401
[25] FAISS: A library for efficient similarity search Johnson et al., 2017 https://arxiv.org/abs/1702.08734
[26] Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks Reimers Gurevych, 2019 https://arxiv.org/abs/1908.10084
[27] BioBERT: a pre-trained biomedical language representation model for biomedical text mining Lee et al., 2020 https://academic.oup.com/bioinformatics/article/36/4/1234/5566506
[28] LLaMA: Open and Efficient Foundation Language Models Touvron et al., 2023 https://arxiv.org/abs/2302.13971
[29] Prompt Engineering for LLMs in Retrieval-Augmented QA OpenAI Cookbook, 2023 https://github.com/openai/openai- cookbook
[30] Minimalism Yields Maximum Results: Deep Learning with Limited Resource Wang, Haoyu.
[31] Yiheng Wang, Hanbin Luo, Weili Fang, An integrated ap- proach for automatic safety inspection in construction: Do- main knowledge with multimodal large language model
[32] Yaneva, V., von Davier, M. (Eds.). (2023). Advancing Natural Language Processing in Educational Assessment (1st ed.). Routledge. https://doi.org/10.4324/9781003278658
[33] Taghavi Far, S.M., Feyzi, F. Large language models for software vulnerability detection: a guide for researchers on models, methods, techniques, datasets, and metrics. Int. J. Inf. Secur. 24, 78 (2025). https://doi.org/10.1007/s10207-025-00992-7
[34] Mohan, R.N.V.J., Raju, B.H.V.S.R.K., Sekhar, V.C., Prasad, T.V.K.P. (Eds.). (2025). Algorithms in Advanced Artificial Intelligence (1st ed.). CRC Press. https://doi.org/10.1201/9781003641537
[35] Campesato, Oswald. "Chapter 5: Fine-Tuning LLMs (1)". Large Language Models for Developers: A Prompt-based Exploration of LLMs, Berlin, Boston: Mercury Learning and Information, 2024, pp. 389-490. https://doi.org/10.1515/9781501520938-007