Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Adaptive Hybrid LLM Framework with Dynamic Model Selection Using Cost and Privacy Constraints

Sangeeta Manna (nee Datta)
BSc. Hons. (Cal.), Master of Computer Applications (UTech. WB. India)
Senior Software Programmer in Banking, Insurance & Service-related IT Industry

Abstract - The integration of Large Language Models (LLMs) into enterprise applications presents critical challenges in achieving an optimal balance between computational performance, operational expenditure, and data confidentiality. Cloud-based LLM services, including those provided via API platforms, deliver superior accuracy and scalability but introduce continuous usage costs and potential exposure of sensitive data. Conversely, locally hosted models offer enhanced control over data and eliminate recurring costs, although they are often constrained by hardware limitations and reduced processing efficiency. This study introduces a hybrid LLM framework that dynamically determines the execution environment based on query sensitivity, token characteristics, and cost thresholds. The proposed system incorporates Retrieval-Augmented Generation (RAG) supported by vector databases such as Pinecone to improve contextual relevance. Additionally, the research examines token-level processing, transformer execution behaviour, and the influence of temperature on response generation. Experimental observations indicate that adaptive routing between cloud and local models significantly enhances efficiency while maintaining privacy and reducing operational costs, making it a viable architecture for enterprise AI systems.

Keywords - Hybrid LLM, RAG, Pinecone, Ollama, OpenAI, Mistral, Cost Optimization, Privacy-Aware Systems

I. INTRODUCTION

The rapid advancement of Large Language Models has transformed the landscape of enterprise software systems by enabling automation, intelligent querying, and enhanced decision support. Organizations increasingly rely on these models to process large volumes of unstructured data and generate meaningful insights. Cloud-hosted LLMs provide immediate access to powerful models with minimal infrastructure requirements. Their ability to process large token sequences efficiently makes them highly suitable for complex tasks. However, their dependency on external infrastructure introduces concerns related to cost scalability and data governance. Alternatively, locally deployed models offer a controlled environment where sensitive information remains within the organizational boundary. While this approach addresses privacy concerns and reduces operational costs, it often struggles to match the performance levels of cloud-based systems due to limited computational resources. These contrasting characteristics highlight the necessity for a hybrid solution capable of dynamically leveraging both environments. This research focuses on developing such a system, where model selection is guided by token characteristics, data sensitivity, and cost considerations. Unlike existing hybrid LLM approaches that primarily focus on cost and deployment strategies, the proposed framework introduces token-aware and temperature-aware decision mechanisms. This enables fine-grained control over model selection, improving both efficiency and response reliability in enterprise environments

II. AIMS AND OBJECTIVES

The primary aim of this study is to design and evaluate a hybrid LLM framework that optimizes performance, cost efficiency, and data privacy.

The objectives of this study include:

- Designing a Retrieval-Augmented Generation (RAG)-based architecture [1]
- Implementing cloud-based interaction using OpenAI models
- Deploying local LLMs using Mistral via Ollama
- Analysing token-level processing behaviour
- Evaluating system performance across multiple parameters
- Proposing a dynamic decision engine for model selection

III. PROBLEM STATEMENT

Selecting an appropriate deployment strategy for Large Language Models remains a complex challenge for organizations due to competing priorities such as cost efficiency, system performance, and data security. Cloud-based LLM solutions provide high computational power and scalability but rely on a usage-based pricing model that can lead to escalating costs. Furthermore, transmitting sensitive

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

business data to external services introduces risks related to privacy and regulatory compliance. In contrast, locally deployed models ensure better control over data and eliminate external dependencies, but they are often limited by hardware capabilities, resulting in slower processing and reduced scalability. The lack of an adaptive mechanism to balance these trade-offs leads to inefficiencies in enterprise environments. Therefore, there is a need for a unified framework that can intelligently select the most suitable processing approach based on contextual and operational constraints.

IV. METHODOLOGY & DESIGN

A. System Architecture Overview

The proposed system follows an adaptive architecture consisting of the following components:

- Query Analyzer
- Decision Engine
- Vector Database (Pinecone) [5]
- Dual LLM Endpoints (Cloud and Local)

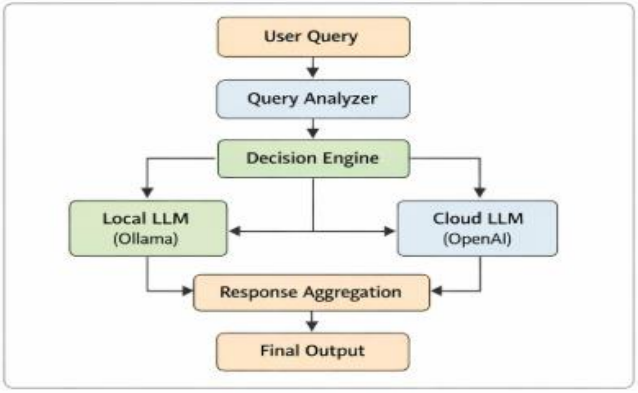

Figure 1: Hybrid LLM Architecture

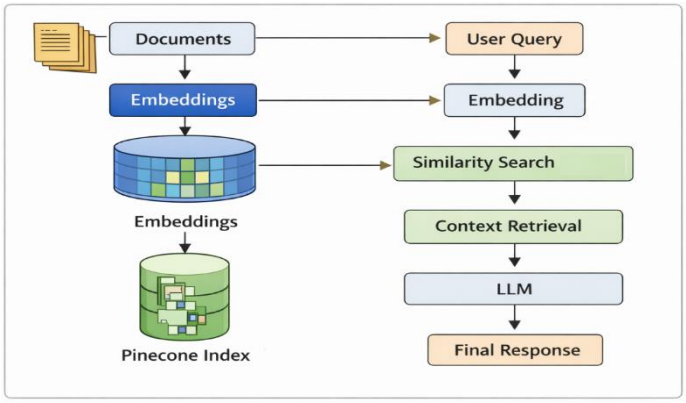

Figure 2: RAG Pipeline with Vector Database

B. Query Classification and Decision Logic

The system classifies incoming queries into two categories:

- Sensitive Queries: Financial, personal, or confidential data
- Non-Sensitive Queries: General informational requests Decision Rules
- Sensitive query -> Process using Local LLM
- High token / performance-critical query -> Process using Cloud LLM
- Cost threshold exceeded -> Process using Local LLM System Benefits
- Ensures data privacy
- Optimizes operational cost
- Maintains high performance

C. Tokenization and Internal Model Execution

1. Scientific Basis of Tokenization

In the proposed hybrid framework, tokenization plays a critical role in enabling efficient query analysis and model selection, where raw text is transformed into structured numerical representations suitable for computational processing. Since neural networks cannot directly interpret human language, tokenization serves as a bridge between linguistic input and mathematical computation.

Modern LLMs use sub word tokenization techniques such as:

- Byte Pair Encoding (BPE)
- WorkPieces
- Unigram Language Model

These approaches decompose words into smaller meaningful units.

Example: ‘Unreachable’ -> [Un] [reach] [able]

This method:

- Reduces vocabulary size
- Handles unknown words efficiently
- Captures linguistic structure

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

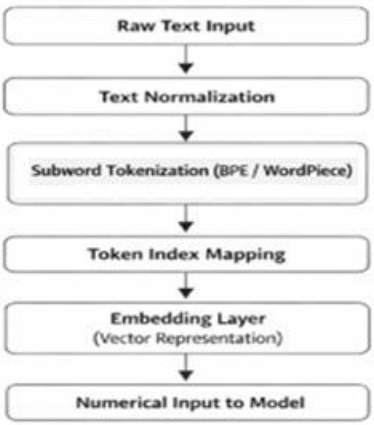

Figure 3: Scientific Tokenization Process

2. Token Processing Flow

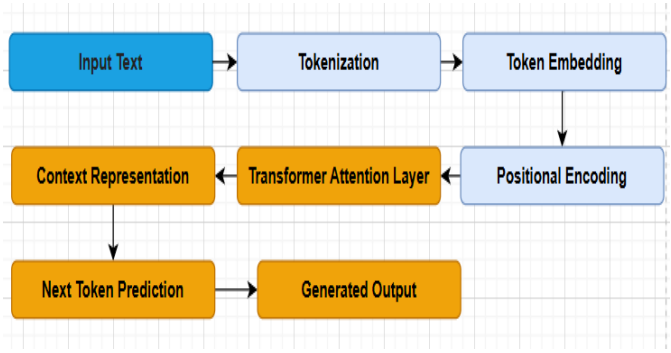

Figure 4: Internal Token Processing in LLM

3. Transformer-Based Execution

LLMs utilize transformer architectures with self-attention mechanisms to capture relationships between tokens. The model generates output using conditional probability:

P(next_token | previous_tokens)

4. Temperature-Based Generation

Temperature is a critical hyperparameter in Large Language Models that directly influences how tokens are selected during response generation. After the model computes logits (raw scores) for all possible next tokens, temperature scaling is applied before converting them into probabilities using the softmax function:

P'(token) = softmax(logits / T)

where **T** represents the temperature value.

a. Effect of Temperature on Token Probability Distribution

Temperature modifies the sharpness of the probability distribution over candidate tokens:

- **Low temperature (T < 0.3)** compresses the distribution, making high-probability tokens significantly more dominant. This leads to deterministic and consistent outputs.
- **High temperature (T > 0.7)** flattens the distribution, increasing the likelihood of selecting lower-probability tokens, thereby introducing variability and creativity.

Thus, temperature acts as a control mechanism that adjusts how confidently the model selects the next token.

b. Relation with Token Generation Process

LLMs generate responses iteratively, one token at a time, based on conditional probability:

P (next_token || previous_tokens)

At each step of generation, temperature influences the selection of the next token. Since each generated token becomes part of the context for subsequent predictions, temperature indirectly affects the entire response sequence.

- Lower temperature ensures stable token sequences with minimal deviation
- Higher temperature increases diversity but may introduce inconsistencies

Therefore, temperature continuously impacts the token generation process throughout the response lifecycle.

c. Impact on Cloud vs Local Models

In cloud-based models such as those provided by OpenAI [2]:

- Large-scale training and optimization enable stable token generation
- Temperature variations have controlled effects
- Even at moderate temperature, coherence is maintained

In local models such as Mistral executed via Ollama [3]:

- Smaller model size increases sensitivity to temperature
- Higher temperature may lead to less coherent outputs
- Careful tuning is required for reliable results

d. Role in Hybrid Decision Framework

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Temperature is incorporated into the hybrid decision system as an additional control parameter alongside token size and query sensitivity:

- **Sensitive or critical queries** -> processed with low temperature to ensure deterministic output
- **General enterprise queries** -> processed with moderate temperature for balanced responses
- **Non-critical or exploratory queries** -> may use higher temperature

This allows the system to dynamically balance accuracy, diversity, and reliability based on context.

e. Link with Enterprise Use-Cases

In enterprise environments, response reliability is more critical than creativity. Therefore:

- Financial, healthcare, and compliance-related queries require **low temperature settings** to ensure accuracy
- Customer support and informational systems can operate with **moderate temperature**
- Analytical or exploratory systems may benefit from **higher temperature settings**

By aligning temperature with use-case requirements, the system ensures optimal performance and controlled output behaviour.

5. Cloud vs Local Processing Behaviour

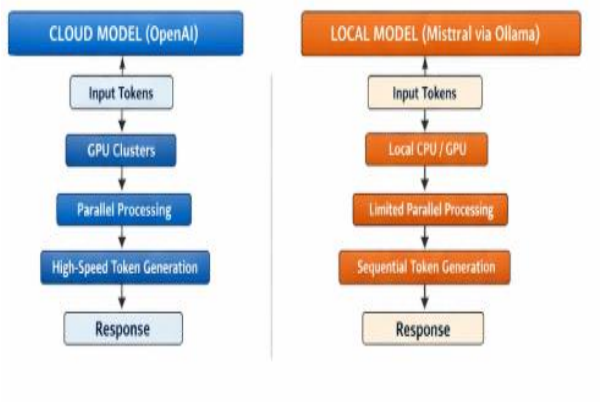

Figure 5: Cloud vs Local Token Processing

Key Insight

- Large token queries -> Cloud processing
- Sensitive queries -> Local processing
- Cost-sensitive scenarios -> Local processing

V. IMPLEMENTATION

The system is implemented as a REST-based backend service that integrates a hybrid decision engine for intelligent query routing. The decision engine applies rule-based logic to evaluate incoming requests based on token size, data sensitivity, and cost constraints before selecting the appropriate execution environment. Cloud-based processing is performed using OpenAI APIs, while local execution is handled through Ollama [4] running the Mistral model. Additionally, the system incorporates Pinecone as a vector database to perform efficient similarity search, ensuring that only relevant contextual information is retrieved and included in the prompt, thereby optimizing token usage and improving response quality.

A. Hybrid Decision Engine Implementation

```
    Edit | Explain | Test | Document | Fix
    public String processQuery(String query) {
        if (isSensitive(query)) {
            return callLocalModel(query);
        } else if (isHighToken(query)) {
            return callCloudModel(query);
        } else {
            return callCloudModel(query);
        }
    }
}
```

A. OpenAI API Integration

```
Edit | Explain | Test | Document | Fix
public static String generateResponse(String prompt) {  1 usage
    try {
        URL url = new URL( spec: "https://api.openai.com/v1/chat/completions");
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();

        conn.setRequestMethod("POST");
        conn.setRequestProperty("Authorization", "Bearer " + API_KEY);
        conn.setRequestProperty("Content-Type", "application/json");
        conn.setDoOutput(true);

        String body = "{ \"model\": \"gpt-4\", \"messages\": [{\"role\":\"user\",\"content\":\"" + prompt +

        try (OutputStream os = conn.getOutputStream()) {
            os.write(body.getBytes());
        }

        BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
        StringBuilder response = new StringBuilder();
        String line;

        while ((line = br.readLine()) != null) {
            response.append(line);
        }

        return response.toString();

    } catch (Exception e) {
        return "Error: " + e.getMessage();
    }
}
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

The vector database is initially implemented using Chroma for local testing and later extended to Pinecone [5] for scalable production deployment. The decision engine is implemented using rule-based logic to ensure simplicity, transparency, and reliability.

VI. RESULT ANALYSIS

The system was evaluated using enterprise-level queries related to financial policies and operational workflows. The results demonstrate clear trade-offs between cloud and local models.

Metric	OpenAI	Mistral	Hybrid
Latency	Low	Medium	Medium
Accuracy	High	Medium	High
Cost	High	Zero	Optimized
Privacy	Low	High	High

The hybrid system significantly reduces API usage while maintaining high accuracy. Sensitive queries are processed locally, ensuring compliance with data protection and privacy requirements.

VII. DISCUSSION

The proposed framework demonstrates that a single-model approach is insufficient for enterprise applications. The hybrid model enables organizations to leverage the strengths of both cloud and local systems. The integration of Pinecone enhances scalability, allowing the system to handle large datasets efficiently. The decision engine ensures that each query is processed using the most appropriate model, improving overall system efficiency.

VIII. FUTURE WORK

Future improvements may include replacing the rule-based decision engine with machine learning models capable of understanding query intent more accurately. Real-time cost monitoring and predictive routing can further optimize system performance. Additionally, integrating multiple LLM providers and improving local model performance through GPU acceleration can enhance scalability and efficiency.

IX. CONCLUSION

This paper presents an adaptive hybrid LLM framework that dynamically selects between cloud-based and local models based on cost and privacy constraints. The integration of RAG and vector databases improves contextual accuracy, while the decision engine ensures optimal model selection.

The results demonstrate that hybrid architectures provide a practical and efficient solution for enterprise AI systems, balancing performance, cost, and data privacy.

REFERENCES

[1] P. Lewis et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” in Advances in Neural Information Processing Systems (NeurIPS), 2020.
DOI: 10.48550/arXiv.2005.11401
[2] OpenAI, “OpenAI API Documentation.”.
Available: https://platform.openai.com/docs
[3] Mistral AI, “Mistral Model Documentation.”.
Available: https://docs.mistral.ai
[4] Ollama, “Ollama Documentation.”.
Available: https://ollama.ai/docs
[5] Pinecone Systems, “Pinecone Vector Database Documentation.”.
Available: https://docs.pinecone.io
[6] A. Vaswani et al., “Attention Is All You Need,”
in Advances in Neural Information Processing Systems, 2017.
DOI: 10.48550/arXiv.1706.03762