LLM Project Report

Amr Yonis
Matriculation Number: 5260015
and **Omar Taher**
Matriculation Number: 5252235

Abstract

Cultural Question Answering (QA) requires models to navigate specific nuances of diverse regions—specifically in our case Iran, China, the United Kingdom, and the United States—that are often underrepresented or generalized in base Large Language Models (LLMs). This report presents a system attempting to solve this task using `Mistral-7B-Instruct-v0.2`. We implement a hybrid architecture combining Parameter-Efficient Fine-Tuning (LoRA) with a dynamic, confidence-aware Retrieval-Augmented Generation (RAG) pipeline.

Our system features a specialized `WikipediaRAG` index filtered by country-specific keywords and a fallback mechanism that triggers real-time web search via DuckDuckGo when internal confidence is low. By employing a "Check-Retrieve-Verify" inference loop, we ensure that external context is only utilized when the model's intrinsic knowledge is insufficient or irrelevant. This approach provides a balance between the speed of direct generation and the factual accuracy of retrieval-based systems.

1 Introduction

Large Language Models (LLMs) have demonstrated exceptional capabilities in general reasoning. However, they are prone to "hallucinations"—generating plausible but factually incorrect assertions—especially when dealing with specific cultural festivals, traditions, or local geography. For the Cultural Commonsense QA task, relying solely on a pre-trained model is insufficient due to the long-tail nature of cultural knowledge.

To address this, we propose a system that integrates three distinct layers of knowledge:

1. **Parametric Knowledge:** Optimized via Low-Rank Adaptation (LoRA) [1] on the provided SAQ and MCQ training datasets.
2. **Static External Knowledge:** A FAISS-indexed subset of Wikipedia, filtered for relevance to the target countries.
3. **Dynamic External Knowledge:** Live web search capabilities to handle queries regarding niche or highly specific entities not found in the static dump.

This report details the implementation of this pipeline, the specific preprocessing strategies employed for cultural data, and the algorithmic logic defining the inference process.

2 Literatute Review

Our system architecture is grounded in recent advancements in efficient large language model (LLM) adaptation and dynamic retrieval-augmented generation. We used **Mistral 7B** [3] as our base model. To adapt this general-purpose model to our specific Short Answer Question (SAQ) and Multiple Choice Question (MCQ) tasks, we implemented **Low-Rank Adaptation (LoRA)** [1]. By injecting trainable rank decomposition matrices into the attention layers while freezing the pre-trained weights, we adhered to the efficiency principles outlined by [2] in their work on QLoRA. This approach significantly reduced memory overhead, enabling us to fine-tune the model effectively.

To mitigate the hallucination issues common in pure parametric models, we integrated a **Retrieval-Augmented Generation (RAG)** framework [4]. Our retrieval pipeline relies on **Sentence-BERT** [8] for generating semantically rich dense embeddings of our Wikipedia knowledge base, and we utilize the **FAISS** library [7] for high-speed similarity search via inner-product indexing.

However, standard RAG can be inefficient if applied indiscriminately. To address this, our inference logic incorporates strategies from **Active RAG (FLARE)** [6] and **Corrective RAG**

(CRAG) [5]. Inspired by FLARE, we implemented a confidence-based thresholding mechanism; if the model's generation confidence—derived from token probabilities—exceeds a set threshold, the system bypasses retrieval entirely, relying solely on parametric knowledge. Conversely, when confidence is low, the system retrieves external context. To ensure the quality of this context, we adopted a corrective mechanism similar to CRAG: a lightweight evaluator checks the relevance of retrieved documents against the query. If the local Wikipedia index fails to provide relevant information, the system dynamically falls back to a web-based search, ensuring the model is grounded in accurate and pertinent data.

3 Methodology

3.1 Data Preprocessing and Prompt Engineering

- **Prompt Formatting:** We utilized `apply_chat_template` to ensure the injection of Mistral-specific control tokens (e.g., `<s>`, `[INST]`) which are critical for maintaining instruction adherence.
- **Defensive Parsing (MCQ):** To handle output entropy, we implemented a regex-based parser. It prioritizes JSON outputs, falls back to standalone letters (e.g., `\b[A-D]\b`), and defaults to "A" only as a last resort.
- **Context-Aware Few-Shotting:** Unlike standard prompting, our system dynamically selects few-shot examples based on the target country (e.g., showing an example question with "Washington DC" as the answer for United States) to prime specific cultural neurons.

3.1.1 SAQ Preprocessing

The SAQ dataset contained raw annotation strings. We implemented an extraction function, `extract_answers`, utilizing `ast.literal_eval` to parse the annotations and retrieve valid English answers ("en_answers"). Rows with empty answers were discarded.

We utilized the `tokenizer.apply_chat_template` method to format inputs, ensuring compatibility with Mistral's instruction-tuning. A country-agnostic few-shot prompt was prepended to all inputs:

> *"Read the following question and provide a single answer without any explanations..."*

This was followed by examples (e.g., "Question: What is the traditional New Year celebration in Iran? Answer: Nowruz") to guide the model toward concise outputs.

3.1.2 MCQ Preprocessing

For the MCQ task, we parsed the JSON-formatted "choices" column. To aid the model, we implemented country-specific few-shot prompting. The `preprocess_mcq_batch` function detects the country code (IR, CN, GB, US) and selects a relevant example from `MCQ_TRAIN_FEW_SHOT`. For instance, questions labeled "CN" (China) are preceded by an example about Beijing, while "US" questions use an example about Washington D.C. This explicitly primes the model for the cultural context of the question.

3.2 LoRA Architecture and Configuration

We employed Low-Rank Adaptation (LoRA) to adapt `Mistral-7B` efficiently, utilizing distinct configurations optimized for the cognitive demands of each task type:

- **Generative Task (SAQ):** Generating full sentences requires higher model capacity. We configured the adapter with Rank $r = 16$ and Alpha $\alpha = 32$. Following QLoRA best practices [2], we targeted all linear layers (`q, k, v, o`) to maximize reasoning capabilities [2]. A dropout of 0.1 was applied to prevent overfitting on the small dataset.
- **Discriminative Task (MCQ):** Since selecting an option (A-D) is a simpler classification task, we reduced the Rank to $r = 8$ and Alpha to $\alpha = 16$. We targeted only the query and value projections (`q, v`) and lowered the dropout to 0.05. This "lighter" configuration reduces memory usage.

3.3 Training Configuration and Strategy

Training was executed using the `Hugging Face Trainer` with `AdamW` optimizer. We implemented specific strategies to handle memory constraints and ensure stability:

Table 1: Comparison of LoRA Configurations for SAQ and MCQ Tasks

Hyperparameter	SAQ	MCQ
LoRA Rank (r)	16	8
LoRA Alpha (α)	32	16
Dropout	0.1	0.05
Target Modules	$q\_proj, k\_proj, v\_proj, o\_proj$	q_proj, v_proj

3.3.1 Hyperparameters

- **Effective Batch Size (EBS):** To maintain mathematical consistency across tasks despite varying input lengths, we normalized the EBS to 16 samples using gradient accumulation:
  - *SAQ:* Batch Size 4 $\times$ Accumulation 4 $= 16$.
  - *MCQ:* Batch Size 2 $\times$ Accumulation 8 $= 16$. (Batch size was reduced here to accommodate the longer 768-token sequence length).
- **Learning Rate Schedule:** We used a learning rate of 2×10^{-4}. For the generative SAQ task, we applied a Cosine Scheduler with a 10% warmup ratio to stabilize the initial adaptation of the new LoRA weights.
- **Epochs:** Both tasks were trained for exactly 3 epochs to prevent the model from memorizing specific phrasing (overfitting). This optimal epoch count was discovered by light cross validation testing.

3.4 Knowledge Retrieval (RAG)

To support the model with external cultural facts, we constructed a specialized "Country-Aware" RAG index [4]. This process involved three distinct stages: heuristic filtering, semantic chunking, and dense vector indexing.

3.4.1 Index Construction & Filtering

We utilized the `wikimedia/wikipedia` dataset as our knowledge source. To ensure high relevance and reduce index noise, we implemented a heuristic tagging system based on keyword frequency.

- **Keyword Dictionaries:** We defined explicit keyword lists for each target country (e.g., Iran: *["nowruz", "farsi", "tehran", ...]*; China: *["dynasty", "confucius", "lunar new year", ...]*).
- **Country Detection:** The `get_article_country` function scans the first 2000 characters of every article. It counts keyword occurrences and assigns the country code (IR, CN, GB, US) with the highest frequency. Articles with no matches are labeled "general" and treated as fallback context.
- **Semantic Chunking:** To fit within the embedding model's context window, articles were segmented into 300-character chunks with a 50-character overlap. This overlap prevents semantic truncation, ensuring that sentences split across boundaries remain intelligible.

3.4.2 Vector Embedding & FAISS

For retrieval, we employed **Dense Vector Search** using the `FAISS` library [7].

- **Embedding Model:** We utilized `sentence-transformers/all-MiniLM-L6-v2` to encode text chunks into 384-dimensional vectors.
- **Normalization:** We applied L2 normalization to all vectors.
- **Indexing:** We instantiated a `faiss.IndexFlatIP`. This "Flat" index performs an exhaustive search, guaranteeing accurate nearest neighbors.

3.5 Dynamic Inference Algorithm

We implemented a *Corrective RAG* pipeline [5] that dynamically switches between internal parametric knowledge, static retrieval, and live web search based on model confidence [6]. This is shown in **Algorithm 1**.

3.5.1 Relevance Verification

A naive RAG system might retrieve irrelevant documents that hallucinate an answer. To mitigate this, we implemented a `check_relevance` function

Algorithm 1 Dynamic RAG Inference Loop

Input: Question Q, Threshold τ, Model M
A_{direct}, conf $\leftarrow$ Generate(M, Q)
if conf $\geq \tau$ **then**
    return A_{direct}
end if
$C_{wiki} \leftarrow$ Retrieve$(Q, \text{top_k} = 3)$
$Relevant \leftarrow$ CheckRelevance(M, Q, C_{wiki})
if not $Relevant$ **then**
    $Q_{search} \leftarrow$ GenerateKeywords(M, Q)
    $C_{web} \leftarrow$ DuckDuckGo(Q_{search})
    $C_{final} \leftarrow C_{web}$
else
    $C_{final} \leftarrow C_{wiki}$
end if
$A_{rag} \leftarrow$ Generate(M, Q, C_{final})
return A_{rag}

acting as a gatekeeper. We prompt the LLM with the retrieved context and the question, explicitly instructing:

> *"Does the provided context contain the answer to the question? Answer ONLY with 'YES' or 'NO'."*

We parse the output tokens; if "YES" is not found, the static context is discarded, triggering the web search fallback. This is also shown in **Figure 1**.

3.5.2 Keyword Generation & Web Search

When the static Wikipedia index fails (e.g., for very niche or recent queries), the system pivots to the live web.

- **Query Formulation:** We do not search with the raw user question (which may be verbose). Instead, the function `generate_search_query` prompts the fine-tuned model to extract essential keywords (e.g., "What is the capital of China?" $\rightarrow$ "capital, China"). We also added a heuristic that appends the target country name if it is missing from the generated keywords to.
- **DuckDuckGo Integration:** We utilize the `ddgs` library for privacy-preserving search. The `perform_web_search` function includes a robustness check: it first attempts to fetch results via the API backend and automatically falls back to the HTML backend if the API is rate-limited or fails, ensuring high availability during inference.

3.6 Evaluation

We evaluated the model on the provided test datasets. Post-processing was applied to SAQ outputs using the `normalize_answer` function, which removes punctuation, lowercases text, and strips articles ("the", "a") to ensure cleaner matching. For MCQ, we used regex matching in `extract_mcq_choice` to robustly identify the predicted letter (A, B, C, or D).

4 Results and Analysis

4.1 RAG Utilization Rates

Our dynamic thresholding mechanism successfully differentiated between "known" and "unknown" queries. For SAQ, the RAG pipeline was triggered in approximately **40.9%** of cases. For MCQ, retrieval was triggered in **8.3%** of cases. This was despite setting a higher threshold for MCQ. This suggests that the model had much higher confidence for the MCQ task due to it's more limited scope than SAQ. With only 4 possible answers, the confidence distrubtion was much less spread out.

4.2 Qualitative Analysis

Effectiveness of Country-Specific Prompts: We observed that explicitly mentioning the country context (e.g., "This question is about Iran") in the MCQ prompt significantly reduced ambiguity. For example, questions about "New Year" were correctly identified as "Nowruz" for Iran versus "Spring Festival" for China, preventing cultural crossover errors.

Web Search vs. Static Index: The static Wikipedia index provided high-quality context for historical facts. However, the Web Search fallback was essential for modern queries or specific entity attributes not present in the simplified Wikipedia dump. The keyword extraction step proved helpful, converting complex natural language questions into effective search strings. We should note that our approach remains incomplete. Much more tuning could be done especially for the RAG retrieval. In out testing, some of the retrieved documents were quite irrelevant leading us to believe further tuning could have been done. Unfortunately due to time constraints and repeating HPC outages, along with submission platform issues, these modifications were not implemented.

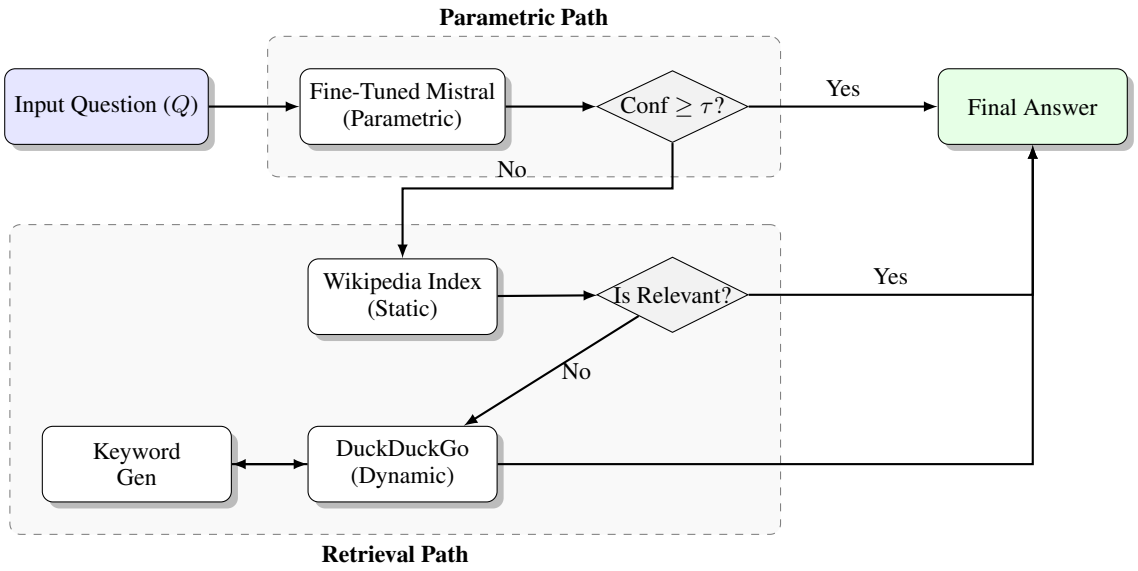

Figure 1: Architecture of the Hybrid RAG System. The system prioritizes parametric knowledge. If confidence is low, it falls back to a static Wikipedia index. If retrieved documents are irrelevant, it triggers a dynamic keyword generation and web search pipeline.

4.3 Numerical Results

Our final results were 77% accuracy for the MCQ task and 64% for the SAQ task giving us a total average of 70%. This highlights how harder it is for the model to generate responses to open ended questions as opposed to making a choise out of 4 answers. In addition to strict word matching rules, this resulted in lower accuracy for the SAQ task.

5 Conclusion

This project demonstrates a method for enhancing Cultural Commonsense QA in 7B-parameter models. By fine-tuning with LoRA and wrapping the model in a confidence-aware retrieval loop, we achieved a system that is both efficient (answering easy questions directly) and knowledgeable (retrieving data for hard questions). The inclusion of a relevance check and web search fallback further protects the pipeline against hallucinations and incomplete information.

References

[1] Edward J. Hu, Yelong Shen, et al. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *ArXiv*, abs/2106.09685.

[2] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. *Advances in Neural Information Processing Systems*, 36.

[3] Albert Q. Jiang, Alexandre Sablayrolles, et al. 2023. Mistral 7B. *ArXiv*, abs/2310.06825.

[4] Patrick Lewis, Ethan Perez, et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

[5] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective Retrieval Augmented Generation. *arXiv preprint arXiv:2401.15884*.

[6] Zhengbao Jiang, Frank F. Xu, Luyu Gao, et al. 2023. Active Retrieval Augmented Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992.

[7] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*.

[8] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

A Appendix

The core logic of the system is implemented in `main.ipynb`. The code can be found in this repository. LLM prompting was used for syntax completion and helper function generation. We submitted to the leaderboards under the username "amryonis".