SecretLLM Project Report

Aleksei Buvailik
TU Dresden
Matriculation number: 5271683
Codabench nickname: albu670g
`email@domain.tld`

Abstract

This report tracks my iterative development of a QA system built around LoRA adapters (Hu et al., 2021), structured prompting, and retrieval-augmented context (Lewis et al., 2020). I detail how each change affected MCQ and SAQ accuracy, including parsing refinements, logprob scoring, and RAG variants. The final results summarize the best combined configuration and the limits of the current approach.

1 Introduction

I develop a QA pipeline for multiple-choice (MCQ) and short-answer (SAQ) tasks that require concise, format-constrained responses. The system adapts a base LLM using parameter-efficient LoRA adapters (Hu et al., 2021) and evaluates both selection-based and generative answering. I also explore retrieval augmentation for SAQ (Lewis et al., 2020), aiming to improve factual recall without changing the base model.

The report has three goals: (1) describe the end-to-end pipeline and implementation choices; (2) document iterative improvements grounded in the experiment logs; and (3) summarize final performance and limitations. The main contributions are:

- a LoRA-based training setup tailored to MCQ and SAQ formatting;
- a log-probability MCQ scorer with a lightweight country prior;
- a BM25-based RAG module with optional preprocessing for SAQ;
- an ablation-style evaluation of prompt, parsing, and retrieval variants.

2 System Overview

The pipeline consists of four stages: data preparation, task-specific LoRA training, inference with validation, and evaluation. MCQ and SAQ datasets are loaded from CSV files, split into training and validation subsets, and tokenized with a task-specific prompt template. Two separate LoRA adapters are trained—one per task—to keep the output formats stable and the adapters lightweight.

During inference, the system routes MCQ questions to a log-probability scorer and routes SAQ questions to a constrained generative path. Both modes share a validation layer that enforces the required output format and triggers limited retries when the response is invalid. For SAQ, retrieval augmentation can be enabled to insert a short context before answering.

The evaluation reports accuracy overall and by country tag for MCQ/SAQ. Section 4 details the iterative experiments, while Section 5 consolidates the final scores.

3 Implementation

The implementation builds on PyTorch and the Hugging Face Transformers stack (Paszke et al., 2019; Wolf et al., 2020), with Hydra used for configuration management (Yadan, 2019). The codebase separates training and inference scripts for clarity and reproducibility.

3.1 LoRA Fine-Tuning

I apply LoRA adapters (Hu et al., 2021) to the attention projection layers (`q_proj`, `k_proj`, `v_proj`, `o_proj`) with rank $r = 16$, $\alpha = 32$, and dropout 0.05. This keeps the number of trainable parameters small while allowing task-specific adaptation. For SAQ, I also test an extended configuration that includes `gate_proj` to increase MLP capacity.

3.2 MCQ Inference via Logprob Scoring

Instead of generating a full answer, MCQ inference scores each option by the log-probability of producing a single-letter continuation. The algorithm performs a forward pass on the prompt to obtain the KV cache and the distribution for the next token. For each choice in {A,B,C,D}, I evaluate several textual variants (“ A”, “\nA”, “A”) and sum token-level log-probabilities using the cached states. The best variant score is selected for that choice, and the highest-scoring choice is returned.

I optionally add a lightweight country-aware prior derived from the MCQ training data. The prior is computed as a smoothed log-probability of the target country tag conditioned on the option text, and scaled by a tunable weight during inference.

3.3 RAG for SAQ

For SAQ, I integrate a retrieval-augmented generation path (Lewis et al., 2020). The retriever builds a Wikipedia-based index using BM25 (Robertson and Zaragoza, 2009) over tokenized document text. Tokenization lowercases, strips punctuation, removes stop words, and applies a lightweight Porter stemmer (Porter, 1980). At inference time, the top-k passages are inserted into a dedicated RAG prompt template. Contexts can also be precomputed and loaded from disk to avoid on-the-fly retrieval.

3.4 Parsing and Validation

The system enforces strict answer formats to stabilize evaluation. SAQ answers must follow a single-line “Answer: <ANSWER>” template with 1–6 tokens, while MCQ responses must include exactly one of {A,B,C,D}. For SAQ, additional regex checks enforce dataset-specific formats (e.g., `HH:MM` or bounded integer ranges). When validation fails, the model is prompted to retry up to a small fixed number of times, and the final response is parsed deterministically.

4 Experiments and Iterative Improvements

This section summarizes the iterative changes evaluated during development. The ordering follows `report/drafts/experiments.md`, and each entry corresponds to a submission and related code changes. I report quantitative results in Section 5; here I focus on the intent and design of each modification.

4.1 Baseline

The baseline uses LoRA adapters on attention projections, standard task prompts, and format validation with limited retries. MCQ uses log-probability scoring, while SAQ uses constrained generation with a strict “Answer:” prefix.

4.2 SAQ Iterations

I evaluate three categories of changes for SAQ: prompt/format refinements, validation retries, and expanded LoRA target layers. These are designed to reduce parsing errors, support multiword answers, and increase adapter capacity without full fine-tuning.

4.3 MCQ Iterations

For MCQ, I test logprob variants and reranking weights. The goal is to stabilize single-letter selection and leverage country priors derived from the training set when options are ambiguous.

4.4 RAG Variants

RAG experiments compare raw retrieval, stop-word filtering, and stemming configurations. I also analyze a later RAG training run and the resulting inference behavior to assess whether retrieval helps SAQ accuracy.

5 Results

This section reports MCQ and SAQ accuracy overall and by country. It also compares the baseline with the best combined configuration. Tables are filled in Step 4 to match the official metrics in `report/drafts/experiments.md`.

6 Discussion and Limitations

The current pipeline is deliberately lightweight, which introduces several limitations. First, the evaluation focuses on accuracy and does not measure calibration or partial credit for near-miss answers. Second, the RAG index is fixed to a Wikipedia corpus and may contain noisy or outdated passages that conflict with the target distribution. Third, strict output formatting reduces noise but can also discard semantically correct responses that violate the schema. Finally, the experiments emphasize targeted changes rather than a full hyperparameter search, so some gains may be left unexplored.

7 Conclusion and Future Work

This section summarizes the contributions and outlines concrete next steps for further model and retrieval improvements.

References

Edward J. Hu, Yelong Shen, Phil Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Lu Wang. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Douwe Kiela, and Sebastian Riedel. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP: System Demonstrations*.

Omry Yadan. 2019. Hydra: A framework for elegantly configuring complex applications. *arXiv preprint arXiv:1909.11942*.