SecretLLM Project Report

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

Abstract

This report tracks the iterative development of a QA system built around LoRA adapters (Hu et al., 2021), structured prompting, and retrieval-augmented context (Lewis et al., 2020). It details 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

This work develops 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. It also explores 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

LoRA adapters (Hu et al., 2021) are applied 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, an extended configuration that includes `gate_proj` is also tested 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}, several textual variants (" A", "\nA", "A") are evaluated and token-level log-probabilities are summed using the cached states. The best variant score is selected for that choice, and the highest-scoring choice is returned.

An optional lightweight country-aware prior is added 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, a retrieval-augmented generation path is integrated (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 the experiment log, and each entry corresponds to a submission and related code changes. Quantitative results are reported in Section 5; this section focuses on the intent and design of each modification.

Change	Task	ΔAccuracy (pp)
Prompt+parsing refinement	SAQ	+8
Validation retries	SAQ	$\sim$0
Add `gate_proj` LoRA targets	SAQ	+1
Logprob scoring with $w = 1.0$	MCQ	+3
Logprob scoring with $w = 1.4$	MCQ	+4
Logprob scoring with $w = 2.0$	MCQ	+5

Table 1: Summary of iterative changes with approximate accuracy deltas (percentage points) relative to the baseline.

4.1 Experimental Setup

Separate LoRA adapters were trained for MCQ and SAQ using the project Hydra configuration. Unless otherwise stated, LoRA uses $r = 16$, $\alpha = 32$, dropout 0.05, and targets attention projections only. MCQ training uses 3 epochs, batch size 4, gradient accumulation 4, learning rate 1×10^{-4}, and a cosine scheduler. SAQ training uses 3 epochs, batch size 4, gradient accumulation 4, learning rate 1×10^{-5}, and a constant scheduler. Inference is deterministic (temperature 0) with a maximum of 16 generated tokens and up to two validation retries.

Metrics are accuracy overall and by country (CN, IR, GB, US) for both tasks. All runs were executed on a single NVIDIA H100 GPU (95,830 MiB), driver 580.65.06, CUDA 13.0.

4.2 Iterative Improvement Summary

Table 1 summarizes the primary changes and their approximate effects on accuracy. This summary anchors the narrative to the experiment log without listing submission identifiers.

4.3 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. Baseline accuracy is 0.74 for MCQ and 0.50 for SAQ (Table 2 and Table 3).

4.4 SAQ Iterations

Three categories of changes were evaluated 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.

Prompt and parsing refinement. The SAQ prompt was updated to enforce a strict single-line

"Answer:" format and parsing was extended to accept multiword answers. This yielded an improvement of approximately 8 percentage points in SAQ accuracy, primarily by reducing formatting errors.

Validation retries. A lightweight retry mechanism was added for invalid SAQ outputs. In practice this affected only four answers in the evaluation set and did not change aggregate accuracy in a meaningful way.

Extended LoRA targets. The SAQ adapter was retrained with `gate_proj` enabled, increasing MLP capacity. This provided a modest additional gain of about 1 percentage point.

4.5 MCQ Iterations

For MCQ, logprob variants and reranking weights were tested. The goal is to stabilize single-letter selection and leverage country priors derived from the training set when options are ambiguous. A weight of $w = 1.4$ improves accuracy by roughly 4 percentage points over the baseline, and $w = 2.0$ provides the best overall MCQ score in the submission series (Table 2).

4.6 RAG Variants

RAG experiments compare raw retrieval, stop-word filtering, and stemming configurations. A later RAG training run and the resulting inference behavior were analyzed to assess whether retrieval helps SAQ accuracy.

Raw retrieval on SAQ reduces accuracy and noticeably increases "idk" outputs. The index was rebuilt with stop-word removal and stemming; the final RAG training run shows a steady loss decrease to about 1.55 by 1.5 epochs and a plateau around 1.46–1.52 by 3 epochs. Validation improves from eval_loss 1.545 to 1.466 and eval_mean_token_accuracy from 0.671 to 0.680, with entropy decreasing from roughly 1.46 to 1.42. The trend is positive but the accuracy gains saturate after about two epochs, suggesting the need to audit generation quality and data coverage.

At inference time, no meaningful SAQ improvement was observed from the current RAG implementation. The last RAG run achieves 0.58 SAQ accuracy overall with country scores 0.51 (CN), 0.63 (GB), 0.50 (IR), and 0.70 (US), which is slightly below the best non-RAG configuration (Table 3). MCQ accuracy remains unchanged at 0.79 because RAG is applied only to SAQ.

MCQ	**Baseline**	**Best**
Overall	0.74	0.79
CN	0.66	0.74
Iran	0.62	0.68
GB	0.90	0.91
US	0.80	0.84

Table 2: MCQ accuracy for the baseline and best combined configuration.

SAQ	**Baseline**	**Best**
Overall	0.50	0.59
CN	0.40	0.57
GB	0.59	0.66
IR	0.38	0.43
US	0.64	0.71

Table 3: SAQ accuracy for the baseline and best combined configuration.

Potential improvements include tuning k and context length, filtering noisy retrievals with a confidence threshold, and experimenting with learned retrievers or reranking to reduce irrelevant context.

5 Results

We evaluate performance on MCQ and SAQ tasks both overall and by country, comparing the baseline model with the best combined configuration (without RAG).

MCQ Performance. The best combined configuration increases overall MCQ accuracy from 0.74 to 0.79 (+0.05 absolute). The largest improvements are observed for CN (+0.08) and IR (+0.06), while gains for US are moderate (+0.04) and minimal for GB (+0.01).

SAQ Performance. For SAQ, overall accuracy rises from 0.50 to 0.59 (+0.09). The largest increase is observed for CN (+0.17), followed by US (+0.07). GB (+0.07) and IR (+0.05) show smaller but consistent improvements.

RAG Configuration. The final RAG-based run achieves 0.58 overall SAQ accuracy, with country-level scores of 0.51 (CN), 0.63 (GB), 0.50 (IR), and 0.70 (US). This is slightly below the best non-RAG configuration. MCQ accuracy remains unchanged at 0.79, as RAG is applied only to SAQ.

Overall, the combined configuration yields consistent gains across most countries, with particularly strong improvements on SAQ for CN and US.

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 work presents a LoRA-based QA pipeline for MCQ and SAQ with strict format control, log-probability scoring for MCQ, and an optional BM25-based RAG component for SAQ. Across the iteration cycle, the largest SAQ gains come from prompt and parsing refinements, with a smaller contribution from expanding LoRA targets to include `gate_proj`. For MCQ, logprob scoring with tuned weighting improves accuracy over the baseline. The best combined configuration reaches 0.79 MCQ accuracy and 0.59 SAQ accuracy, while the current RAG implementation does not yield a measurable SAQ improvement.

Future work should focus on (1) improving retrieval quality with better indexing, reranking, or learned retrievers; (2) analyzing generation errors to separate formatting failures from semantic mistakes; (3) expanding ablations on LoRA targets and hyperparameters; and (4) adding robust reporting of hardware and training-time metrics to improve reproducibility.

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*.