Summary Report

Paper: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks [2]

Team Members: Claas Beger, Cole Breen, Carl-Leander Henneking, Mihir Mishra, Max Whitton

GitHub: `https://github.com/MaxWhitton25/CS4782-Final`

Class: CS 4782/5782 Introduction to Deep Learning

Date: May 12, 2025

---

1 Introduction

Large Language Models (LLMs) have proven to be effective tools across a wide range of natural language processing tasks. However, their ability to recall factual information is often limited by the fixed context window and static knowledge encoded in model parameters during pre-training, leading to gaps in model knowledge and potential for hallucinations. Retrieval-Augmented Generation (RAG) addresses this limitation by combining parametric knowledge from LLMs with non-parametric memory through document retrieval.

This report reproduces key results from the RAG paper, focusing on abstractive question answering and fact verification. Specifically, we evaluate RAG on the abstractive QA task using the `rag-datasets/rag-mini-bioasq` dataset [1], and on fact verification using a subset of the FEVER dataset [3] available through Hugging Face. We report strong improvements on both tasks using a fine-tuned model with the original RAG architecture, with evaluation measured by BLEU/ROUGE and accuracy, respectively.

2 Chosen Result

For this project, we chose to fine-tune the RAG-Sequence model with Fast Decoding on the FEVER and rag-mini-bioasq datasets. RAG-Sequence was chosen over RAG-Token as it has similar performance and can be fine-tuned faster, enabling multiple training runs with limited access to computational resources. Similarly, Fast decoding was chosen over thorough decoding as performance is similar, and it is more computationally efficient. Both the RAG-Sequence and the Fast Decoding decisions were made to maximize our time developing the actual model and minimize the time spent fine-tuning while ensuring we would still be able to reproduce significant results. The result we are trying to recreate is the improved performance of RAG over baseline (BART) generation: we hope to demonstrate an increase in BLEU/ROUGE score on abstractive QA and an increase in accuracy on the FEVER task.

3 Methodology

Given an input query, the RAG model uses a retriever to obtain the relevant documents and then feeds those documents into a generator, which produces a response to the query conditioned on the retrieved documents.

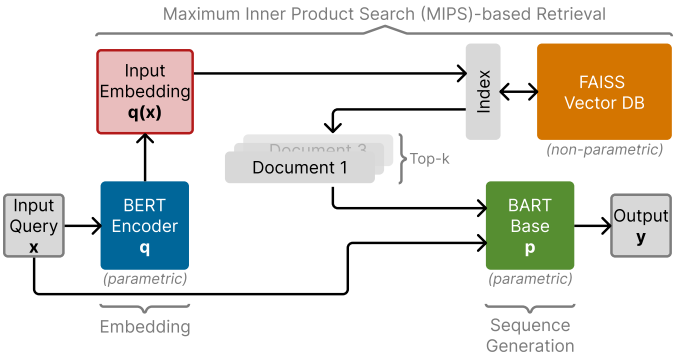

Figure 1: Diagram of the RAG architecture. Includes retrieval and generation flow through all major components starting from input x to the generated output y.

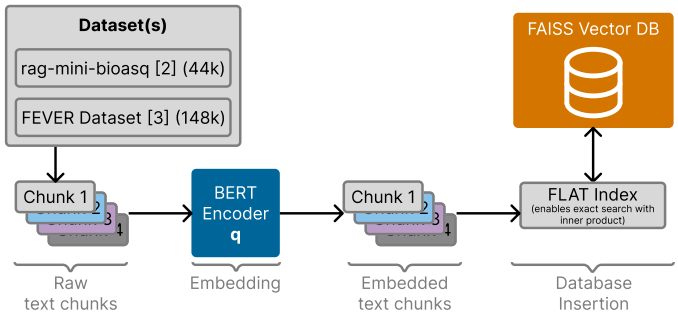

Figure 2: FAISS Vector Database initialization. Color coding matches that of Figure 1 to indicate that the same components are being used.

3.1 Retriever

We use Dense Passage Retrieval (DPR) as the non-parametric memory component. DPR consists of two separate BERT-based encoders: one for the query and one for the documents. The document encoder first transforms the dataset's text corpus into dense vectors, which are stored using Facebook AI Similarity Search (FAISS). During retrieval, the query is encoded separately and then we apply an inner product similarity search between the query vector and the pre-computed document vectors to determine the top-k relevant passages.

3.2 Generator

We use BART-base, an encoder-decoder model with around 140M parameters, for our generator. During generation, the raw text query is concatenated with each retrieved document and passed into BART, which then produces a response conditioned on the query and each document. The model outputs one response for each document, and then (in our implementation) Fast Decoding decides which of these responses will be the final RAG output by taking the argmax of $p(y \mid x)$, with

$$p(y \mid x) \approx \sum_{z \in Z_y} p(z \mid x)\, p(y \mid x, z), \quad \text{where } y \in Y_z.$$

Here, Y_z is the set of sequences generated by beam search for each document z and $Z_y = \{z : y \in Y_z\}$. We assume $p(y \mid x, z) \approx 0$ for any y not in Y_z.

3.3 Training

We fine-tuned the BERT query encoder and the BART generator using a cross-entropy loss on the `rag-mini-bioasq` dataset. Note that the BERT document encoder was not fine-tuned since it would require the entire corpus index to be periodically updated during training, which would be very costly. We fine-tuned the models on our own GPU for 5 epochs. We trained RAG-Sequence with k=3, 5, and 10, and evaluated its performance on the `rag-mini-bioasq`. We also evaluate an alternative use case with an adjusted classification loss on fact verification using the `FEVERV1` dataset.

4 Results & Analysis

Table 1: QA Performance on bio-asq test split for different k training values. For RAG, k is set at train time, and all evaluations are run with k = 1, i.e., one document is retrieved to answer the query.

Method	k	Avg BLEU-1	Avg ROUGE-L
Baseline	-	0.1086	0.2225
Our RAG	1	0.4355	0.3860
Our RAG	3	0.4260	0.3863
Our RAG	5	0.4340	0.3865
Our RAG	10	0.4355	0.3860

For our evaluation of abstractive QA on the `rag-datasets/rag-mini-bioasq` dataset [1], we used BLEU-1 and ROUGE-L as these were what the paper used. For this particular dataset, these are applicable as opposed to exact match because the questions are about the biomedical field and often warrant synthesis of multiple factual points into coherent sentences: this aligns with abstractive QA and overlap-based evaluation metrics. Our results demonstrate a nearly 4x increase in BLEU-1 score and a 1.5x increase in ROUGE-L (for all k values tested) as compared to our raw BART generation baseline. This increase aligns with the finding from the original paper, but represents an even more extreme improvement. We suspect this is related to the dataset question-answer complexity that makes RAG particularly effective. Interestingly, the optimal k value at test time, regardless of the train time k value, was k = 1. Due to the nature of the dataset, we suspect this is because fetching too many documents at test time distracts our model.

Table 2: Retrieval Augmented Generation increases the F1-Score significantly when evaluated on the fact verification dataset FEVER [3]. In line with previous results, we fix k=1.

Approach	Accuracy	Macro-Precision	Macro-Recall	Macro-F1
Baseline	0.2380	0.0793	0.3333	0.1282
RAG	0.6562	0.6626	0.6354	0.6372

We further evaluate on the task of Fact Verification, with a slightly modified evaluation because it is a classification tasks using ACCEPT/REFUTE/UNKNOWN. As Table 4 shows, RAG can improve performance similarly to openQA, with an overall accuracy improvement of 0.4 and an F1 improvement of 0.5. We observe similar trends to openQA and thus keep our efforts focused on k=1.

5 Reflections

Throughout this process, we learned a number of valuable lessons and had some key takeaways from the re-implementation process. As in the original paper, we found that RAG greatly improves model performance in a question-answer and fact verification context. A more specific finding we had, beyond the scope of the original paper, was that it massively improves performance when operating on a niche dataset characterized by highly specialized terminology, concepts, and questions.

We found that modularity of architecture was vital to ensure ease of debugging and comprehensibility, especially when working together as a team on a complex multi-step machine learning pipeline. Additionally, we learned to navigate the

challenges of computing loss when marginalizing over multiple documents, which required careful tracking of query-document pairs and their associated outputs. Furthermore, we learned to infer and synthesize things not explicitly mentioned in the paper, such as selecting a sufficient learning rate, which ensured the stability of training.

Future extensions of this work, beyond what we have done here, would likely involve experimenting with larger models, testing with larger datasets, and performing semantic evaluation of our models. Furthermore, the original paper mentions that fine-tuning the document encoder is costly and didn't yield significant improvements, but we would be interested to see if there are methods to cost-effectively improve performance in certain contexts via doing so. Finally, while not directly related, we also think there is potential for RAG in a multimodal setting: perhaps in the image generation space, conditioning image generation not only on text but also retrieved images would have a similar effect to RAG for text generation.

References

[1] Hugging Face RAG Datasets Community. rag-mini-bioasq. `https://huggingface.co/datasets/rag-datasets/rag-mini-bioasq`, 2024.

[2] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[3] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In *NAACL-HLT*, 2018.