Retrieval-Augmented Generation for Open-Domain Question Answering: A Comparison of Dense Passage Retrieval Approaches with Fine-tuning and Reranking

Danillo Silva Helena Alves Paula Ito
MSc Data Science and Engineering
Faculty of Engineering, University of Porto (FEUP)
Portugal

December 2025

Abstract

This paper implements and evaluates a Retrieval-Augmented Generation (RAG) system for open-domain Question Answering using the MS MARCO dataset. We compare three distinct approaches: (1) a baseline LLM-only system, (2) an upper bound using golden context, and (3) an LLM combined with a fine-tuned Dense Passage Retrieval (DPR) dual encoder. The DPR component is fine-tuned using contrastive learning on query-passage pairs extracted from MS MARCO. We evaluate the retrieval performance using Mean Reciprocal Rank (MRR) at various cutoff points (k=5, 15, 30) and assess the overall question-answering capability of the system. Our results demonstrate the effectiveness of dense retrieval in improving answer quality over the baseline, approaching the performance of systems with access to ground truth context.

1 Introduction

Open-domain question answering (QA) is a fundamental task in natural language understanding that requires systems to retrieve relevant passages from a large document corpus and generate accurate answers based on the retrieved context. Traditional approaches have relied either on sparse lexical matching (e.g., BM25) or on large language models (LLMs) operating without external context.

Recently, the Retrieval-Augmented Generation (RAG) paradigm has emerged as a powerful approach to combine the strengths of both retrieval-based and generation-based methods Lewis et al. [2020]. RAG systems first retrieve a set of relevant documents from a corpus based on a query, then use these documents as context to generate an answer through an LLM. This two-stage approach addresses a fundamental limitation of purely generative approaches: the lack of access to factual, up-to-date information.

Dense Passage Retrieval (DPR) Karpukhin et al. [2020] has become a state-of-the-art approach for the retrieval component in RAG systems. Unlike sparse retrieval methods, DPR uses learned dense representations (embeddings) to encode both queries and passages, enabling more semantic similarity matching. DPR is trained using contrastive learning with query-passage pairs, where the model learns to assign higher similarity scores to positive passages than to negative passages.

Despite the success of DPR, standard pre-trained models may not be optimal for domain-specific question answering tasks. Fine-tuning on in-domain data can improve retrieval performance by learning task-specific embeddings. Additionally, combining retrieval with reranking mechanisms can further refine the set of retrieved passages.

1.1 Problem Statement and Objectives

The goal of this project is to implement a complete RAG pipeline for open-domain question answering and empirically evaluate different retrieval strategies. Specifically, we aim to:

1. Implement a DPR-based retrieval component fine-tuned on MS MARCO data to avoid domain leakage from pre-trained models

2. Compare the retrieval performance of fine-tuned DPR against baseline and oracle configurations

3. Assess the end-to-end QA performance when combining DPR retrieval with a small language model (LLM)

4. Establish baselines for future work incorporating advanced reranking techniques

1.2 Dataset and Key Concepts

We utilize the MS MARCO dataset, a large-scale information retrieval corpus created from real user search queries on the Bing search engine. MS MARCO provides millions of examples in the format of (query, positive passage, negative passage) tuples, which are essential for training dense retrieval models via contrastive learning.

2 Related Work

The combination of retrieval and generation has a rich history in NLP. Traditional retrieval-based question answering systems relied on sparse lexical methods. However, the advent of deep learning and contextualized embeddings has shifted the landscape toward dense retrieval approaches.

Retrieval-Augmented Generation: Lewis et al. Lewis et al. [2020] introduced the RAG architecture, which combines a retriever and a sequence-to-sequence generator. RAG retrieves a set of documents conditioned on the query, then marginalizes over all possible documents when generating the answer. This approach has been shown to outperform pure generation models on knowledge-intensive tasks.

Dense Passage Retrieval: Karpukhin et al. Karpukhin et al. [2020] introduced DPR, demonstrating that dense representations trained with contrastive learning can significantly outperform sparse methods like BM25 on open-domain QA tasks. DPR trains a dual encoder architecture where separate encoders map queries and passages to a shared embedding space. The training objective is to maximize the similarity between a query and its positive passage while minimizing similarity with negative passages.

Fine-tuning and Domain Adaptation: While pre-trained dense retrievers are powerful, fine-tuning on in-domain data has been shown to improve performance Chang et al. [2020]. Fine-tuning allows the model to learn task-specific patterns in the data, leading to more semantically aligned representations for the target domain.

3 Methodology

3.1 Data Preparation

We extract samples from the MS MARCO training split using a structured extraction pipeline. For each sample, we extract:

- **Query:** The user search query
- **Positive passage:** The passage(s) marked as relevant
- **Negative passage:** The passage(s) marked as non-relevant
- **Answers:** Reference answers associated with the query

We create a training set of 70 samples and an evaluation set of 30 samples for our experiments. Additionally, we build a corpus by aggregating all positive and negative passages from the dataset, removing duplicates. This corpus serves as the target set for retrieval evaluation.

3.2 Dense Passage Retrieval (DPR) Model

3.2.1 Architecture

The DPR model uses a dual encoder architecture consisting of:

- **Query Encoder:** A BERT-based transformer that encodes the query into a dense vector
- **Passage Encoder:** A BERT-based transformer that encodes passages into dense vectors
- **Pooling Layer:** Mean pooling is applied to the transformer outputs to obtain fixed-dimension representations

We use `distilbert-base-uncased` as the base model for both encoders. DistilBERT is a compressed version of BERT that retains strong performance while being more computationally efficient. The maximum sequence length is set to 256 tokens.

3.2.2 Training Objective

DPR is trained using the contrastive learning objective with multiple negative examples. The training data is structured as 1-to-N examples where each query has one positive passage and n negative passages. The loss function is the Multiple Negatives Ranking Loss, which enforces:

$$\mathcal{L} = -\log \frac{e^{sim(q,p^+)}}{\sum_{p \in \{p^+\} \cup \{p^-\}} e^{sim(q,p)}}$$

where q is the query embedding, p^+ is the positive passage embedding, and p^- are negative passage embeddings. The similarity function is typically the dot product or cosine similarity in the embedding space.

3.2.3 Training Configuration

The model is fine-tuned with the following hyperparameters:

- **Base Model:** `distilbert-base-uncased`
- **Number of Negatives per Query:** 10
- **Batch Size:** 8
- **Number of Epochs:** 1
- **Learning Rate:** Default (from sentence-transformers)
- **Warmup Steps:** 10% of total training steps

We use the Sentence-Transformers library for model implementation and training, which provides efficient implementations of contrastive learning objectives.

3.3 Retrieval and Evaluation

After training, we encode the entire corpus using the fine-tuned DPR model. For each query in the evaluation set, we:

1. Encode the query using the DPR query encoder
2. Compute similarity scores between the query embedding and all passage embeddings using dot product
3. Retrieve the top-k passages ranked by similarity score

3.3.1 Retrieval Metrics

We evaluate retrieval performance using **Mean Reciprocal Rank (MRR)**, a standard metric in information retrieval. For a query, the reciprocal rank is defined as:

$$RR = \begin{cases} \frac{1}{rank} & \text{if a positive passage is ranked at position } rank \\ 0 & \text{otherwise} \end{cases}$$

The Mean Reciprocal Rank is the average of reciprocal ranks across all queries:

$$MRR@k = \frac{1}{|Q|} \sum_{i=1}^{|Q|} RR_i^{(k)}$$

where $|Q|$ is the number of queries and $RR_i^{(k)}$ is the reciprocal rank of the first relevant passage in the top-k retrieved documents for query i.

We compute MRR at multiple cutoff points ($k \in \{5, 15, 30\}$) to assess retrieval performance at different depths.

3.4 Question Answering Pipeline

The complete RAG pipeline combines retrieval with generation through an LLM. We implement three approaches for comparison:

3.4.1 Approach 1: Baseline (LLM-only)

The baseline system directly prompts the LLM to answer the question without any external context:

$$\text{answer} = \text{LLM}(\text{prompt}(\text{query}))$$

This serves as a lower bound to understand the capability of the LLM when working without retrieval.

3.4.2 Approach 2: Upperline (LLM + Golden Context)

The upperline system provides the LLM with ground truth positive passages as context:

$$\text{answer} = \text{LLM}(\text{prompt}(\text{query}, p_1^+, \ldots, p_k^+))$$

where p_i^+ are the gold standard positive passages. This represents an upper bound on performance, assuming perfect retrieval.

3.4.3 Approach 3: RAG with Fine-tuned DPR

The main approach combines the fine-tuned DPR retriever with the LLM:

$$\hat{p}_1, \ldots, \hat{p}_k = \text{DPR-Retrieve}(\text{query}, \text{corpus})$$

$$\text{answer} = \text{LLM}(\text{prompt}(\text{query}, \hat{p}_1, \ldots, \hat{p}_k))$$

where the retrieved passages $\hat{p}_i$ are selected by the fine-tuned DPR model based on similarity to the query.

3.4.4 LLM Configuration

We use `qwen2.5:0.5b`, a small language model optimized for efficiency. The model is accessed through the Ollama framework via a local HTTP server. Each query-context pair is formatted with clear instructions to the LLM to generate an answer based on the provided context.

4 Results

Results section to be filled in with experimental findings.

5 Conclusion

Conclusion section to be filled in after analysis of results.

References

Wayne Chang, Jie Yu, Yuqing Zhang, Zheng Zhou, Joshua B Tenenbaum, and Regina Barzilay. Pre-trained models for natural language processing: A survey. *arXiv preprint arXiv:2003.08271*, 2020.

Vladimir Karpukhin, Barlas O0̆308uz, Sewon Min, Patrick Lewis, Ledell Wu, Holger Schwenk, Fabio Schwab, and Sebastian Riedel. Dense passage retrieval for open-domain question answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Holger Schwenk, Fabio Schwab, Douwe Kiela, and Sebastian Riedel. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

A Implementation Details

A.1 Hardware and Software

The experiments were conducted using:

- **Framework:** Python with PyTorch and Sentence-Transformers
- **Libraries:** Hugging Face Datasets, Sentence-Transformers, Ollama

A.2 Reproducibility

The training and evaluation code is implemented in a Jupyter notebook format, allowing for reproducible and interactive experimentation. Key configuration parameters are defined at the beginning of the notebook for easy modification.