Reproducibility Report: Evaluating Retrieval Quality in Retrieval-Augmented Generation (eRAG)

Fabiano Mangini
Politecnico di Bari
`f.mangini4@studenti.poliba.it`

April 11, 2025

Course: Information Retrieval and Personalization

Abstract

Evaluating Retrieval-Augmented Generation (RAG) systems, particularly the contribution of the retriever component, presents significant challenges. Traditional end-to-end methods are computationally expensive and lack granular insight, while standard retrieval metrics often show poor correlation with the RAG system's final task performance. The paper "Evaluating Retrieval Quality in Retrieval-Augmented Generation" by Salemi & Zamani proposes eRAG, a novel approach where the downstream Large Language Model (LLM) evaluates the utility of each retrieved document individually against the task's ground truth. This method claims to offer higher correlation with end-to-end RAG performance and improved computational efficiency. This report details a project aimed at reproducing the key findings presented in the eRAG paper, with a primary focus on verifying the claimed correlation between eRAG evaluation scores and the downstream task performance. The reproduction effort targets experiments on the Natural Questions (NQ) dataset from the KILT benchmark, employing BM25 and Contriever as retrievers and a T5-Small model with Fusion-in-Decoder (FiD) as the generator. As the original authors only provided the core 'erag.eval' function, the complete RAG pipeline, including data preprocessing, BM25/Contriever indexing and retrieval, T5-FiD fine-tuning, and evaluation, was reconstructed from scratch using libraries such as Transformers, Faiss, rank_bm25, and the 'erag' package within a Google Colab environment. Significant challenges were encountered, most notably the hardware limitations of using a T4 GPU compared to the A100 used in the original study, impacting model fine-tuning capabilities, and the practicalities of managing the large Wikipedia corpus. This project remains a work in progress. Preliminary experiments conducted on a small corpus subset ($\approx 1/1000$) with limited fine-tuning (1 epoch) on limited data (`nq-dev-kilt.jsonl`) yielded statistically significant but lower correlations (Kendall's Tau ≈ 0.23) than reported, alongside extremely low absolute task performance ($\approx 1\%$ end-to-end Exact Match), likely reflecting these constraints and indicating areas requiring further investigation and resources.

1 Introduction

1.1 The Rise of Retrieval-Augmented Generation (RAG)

In recent years, Large Language Models (LLMs) have demonstrated remarkable capabilities in various natural language processing tasks. However, they often struggle with generating factually accurate responses, especially for knowledge-intensive queries, and can produce "hallucinations" – plausible but incorrect information [1, 28]. Retrieval-Augmented Generation (RAG) has emerged as a powerful paradigm to mitigate these issues [32]. RAG systems typically operate in two stages: first, a retriever module searches a large external knowledge corpus (like Wikipedia) to find documents relevant to an input query; second, a generator module (an LLM) synthesizes an answer based on both the original query and the content of the retrieved documents. This architecture allows RAG systems to ground their outputs in external evidence, reducing hallucination [1, 28], enabling knowledge grounding [10, 16, 31], and facilitating personalization [24, 25].

1.2 Challenges in Evaluating RAG Systems

While RAG systems offer significant advantages, evaluating their performance, particularly the effectiveness of the retrieval component within the overall pipeline, presents unique challenges. Traditional end-to-end evaluation, which compares the final generated output against a ground truth reference [20], provides a holistic measure but suffers from several limitations. It is often computationally expensive, especially when dealing with long contexts formed by multiple documents. Furthermore, end-to-end evaluation lacks transparency; it doesn't reveal *which* retrieved documents were most useful or detrimental to the final output, making it difficult to interpret system behavior or optimize the retriever. This list-level feedback is often insufficient for fine-tuning ranking models, which typically benefit from document-level signals [3, 7].

Alternatively, one might evaluate the retriever using standard information retrieval (IR) metrics based on pre-defined relevance labels (e.g., provenance information indicating which document contains the answer). However, as investigated by Salemi & Zamani and confirmed in their work [27], such relevance labels, often based on human judgment of topical relevance, exhibit surprisingly low correlation with the actual downstream performance of the RAG system. A document deemed relevant by a human might not contain the specific information needed by the LLM, or might present it in a way the LLM cannot effectively utilize.

1.3 The eRAG Paper by Salemi & Zamani

Addressing these evaluation challenges, Salemi and Zamani recently proposed eRAG, a novel approach detailed in their paper "Evaluating Retrieval Quality in Retrieval-Augmented Generation" [27]. eRAG shifts the evaluation focus to the utility of retrieved documents specifically for the *downstream generation task*, as perceived by the LLM itself. The core idea is to use the same LLM from the RAG system as an "annotator". For a given query and its ranked list of retrieved documents, eRAG runs the LLM generator *individually* on the query paired with *each single document*. The output generated from each document is then evaluated against the downstream task's ground truth using the task-specific metric (e.g., Exact Match, F1). This score effectively serves as a measure of that specific document's contribution or relevance *to the task*. These document-level scores can then be aggregated using standard set-based or ranking metrics (like Precision@k or NDCG@k) to provide an overall score for the retrieved list.

1.4 Project Goal and Scope

The primary objective of this project is to conduct a reproducibility study of the eRAG evaluation method proposed by Salemi and Zamani. We aim to reconstruct the necessary experimental pipeline and verify the key claims made in their paper. Specifically, our goals include:

1. Verifying the central claim that eRAG evaluation scores exhibit a higher correlation (measured by Kendall's Tau and Spearman's Rho) with the end-to-end performance of the RAG system compared to baseline evaluation methods (like using KILT provenance labels).

2. Investigating how this correlation changes as the number of retrieved documents ('k') provided to the generator varies (initially planned, pending resource availability).

3. Assessing the claimed computational efficiency advantages (in terms of runtime and memory) of eRAG compared to standard end-to-end evaluation (initially planned, pending resource availability).

Due to resource constraints, our initial reproduction efforts focus primarily on the first goal using the Natural Questions (NQ) dataset from the KILT benchmark [20] and the BM25 retriever.

1.5 Report Structure

This report is structured as follows: Section 2 provides a more detailed background on the eRAG methodology and its theoretical advantages. Section 3 outlines the experimental setup used in the original paper. Section

4 details our specific implementation choices and the process undertaken to reproduce the RAG pipeline components. Section 5 presents our preliminary results from limited-scale experiments and discusses the observed discrepancies and challenges. Section 6 summarizes the difficulties encountered and outlines future work required for a more complete reproduction. Finally, Section 7 concludes the report with a summary of findings and the current status of the reproducibility effort.

2 Background: The eRAG Evaluation Method

2.1 Motivation Revisited: Why eRAG?

The evaluation of retrieval models within RAG systems is complicated by the indirect nature of their contribution. The ultimate goal of the retriever is not just to find topically relevant documents, but to provide evidence that the downstream LLM generator can effectively utilize to produce a correct and well-supported final output [32]. Traditional Information Retrieval (IR) metrics, often relying on human judgments of relevance or simple lexical overlap (like checking if a document contains the ground truth answer string [9, 14, 23, 26]), frequently fail to capture this utility accurately. As observed by Salemi & Zamani [27] and other researchers, metrics based on such relevance judgments (like the KILT provenance labels [20]) often show weak correlation with the end-to-end performance of the RAG system. A document might be topically relevant or even contain the answer verbatim, yet be structured in a way that confuses the LLM or lacks the specific context the LLM needs. Conversely, a document not containing the exact answer string might provide crucial background information that aids generation.

Furthermore, end-to-end evaluation, while measuring the final outcome, provides only a single score for the entire list of retrieved documents. This makes it difficult to pinpoint which documents helped or hindered the LLM, hindering interpretability and providing poor signal for optimizing the retriever, which often benefits from document-level feedback [3][7]. eRAG was proposed to bridge this gap by directly assessing the usefulness of *each* retrieved document from the perspective of the downstream LLM generator.

2.2 eRAG Methodology Explained

The eRAG approach evaluates the retriever by leveraging the RAG system's own LLM generator as an arbiter to assign a relevance score to each document in the retrieved list. The process involves three main steps:

1. **Individual Document Processing:** For a given query q and a ranked list R_k containing k documents retrieved by the model R, each document d in R_k is individually fed to the LLM generator M along with the query q. This produces a separate output $y_d = M(q, \{d\})$ for each document.

2. **Downstream Task Evaluation:** Each individual output y_d is then evaluated against the ground truth answer y for the original query q using the appropriate downstream task metric $\mathcal{E}_M$. This task metric could be Exact Match (EM) for question answering, Accuracy for classification (like FEVER), F1 or ROUGE [17] for summarization or dialogue, or any other relevant metric for the specific task. The resulting score serves as the eRAG relevance label $G_q[d]$ for that document d with respect to query q. Formally, as presented in the paper:

$$G_q[d] = \mathcal{E}_M(M(q, \{d\}), y) \quad : \forall d \in R_k$$

   This score $G_q[d]$ quantifies how useful document d was *on its own* in helping the LLM M arrive at the correct answer y.

3. **Aggregation:** Once each document d in the list R_k has an associated relevance score $G_q[d]$, these scores are aggregated using standard set-based or ranking-based IR evaluation metrics $\mathcal{E}_R$ to produce a final score for the retrieval list R_k. The paper explores various aggregation metrics, including:

   - Precision@k (P@k): Average relevance score of the top k documents.
   - Recall@k (R@k)

- Mean Average Precision (MAP)
- Mean Reciprocal Rank (MRR) [2]
- Normalized Discounted Cumulative Gain (NDCG@k) [11]
- Hit Rate@k / Success@k: Checks if at least one document in the top k received a non-zero relevance score (or the maximum score for non-binary labels).

This aggregated score $\mathcal{E}_R(R_k, G_q)$ reflects the overall quality of the retrieved list R_k in terms of its utility to the downstream LLM for the specific task.

2.3 Claimed Advantages

The authors of eRAG claim several advantages for their method over traditional approaches:

- **Higher Correlation:** The primary claim is that eRAG scores demonstrate a significantly higher correlation (measured via Kendall's Tau and Spearman's Rho) with the final end-to-end performance of the RAG system compared to evaluations based on human relevance labels or simple answer presence checks. This suggests eRAG better reflects the retriever's true impact on the RAG pipeline.
- **Computational Efficiency:** Compared to full end-to-end evaluation (which processes k documents concatenated together), eRAG processes each document individually. While this requires k forward passes through the LLM, the quadratic scaling of transformer attention with input length means processing k *short* inputs (query + single doc) is significantly faster and requires substantially less memory (up to 50x less GPU memory reported) than processing one *very long* input (query + k docs) [27]. This is particularly beneficial when dealing with large k or long documents.
- **Document-Level Feedback:** Unlike end-to-end evaluation, eRAG naturally produces a utility score for each individual document. This granular feedback is valuable for diagnosing retriever weaknesses and can potentially be used as a signal for training or fine-tuning retrieval/ranking models.

3 Original Paper's Experimental Setup

This section outlines the experimental configuration reported by Salemi and Zamani in their work [27] to evaluate the eRAG method.

3.1 Datasets and Corpus

- **Datasets:** The experiments were conducted using five knowledge-intensive language task datasets from the KILT benchmark [20]:
  - Natural Questions (NQ) [15] - Open-domain QA
  - TriviaQA [13] - Open-domain QA
  - HotpotQA [30] - Multi-hop QA
  - FEVER [29] - Fact Verification (Classification task)
  - Wizard of Wikipedia (WoW) [4] - Knowledge-grounded Dialogue Generation

  Due to the unavailability of ground truth labels for the official KILT test set, the authors utilized the publicly available *validation sets* for their experiments.
- **Retrieval Corpus:** The knowledge source was the KILT Wikipedia dump (`kilt_knowledgesource .json`, snapshot from 01/08/2019). Following the preprocessing protocol established by Karpukhin et al. [14], each Wikipedia article was segmented into passages, with each passage constrained to a maximum length of 100 words. For retrieval purposes, each document was constructed by concatenating the article title with the passage text, separated by a `[SEP]` token.

3.2 Models

- **Generator LLM:** The primary generator model used throughout the experiments was **T5-Small** [21] integrated with the **Fusion-in-Decoder (FiD)** architecture [10]. For experiments investigating the impact of model size, T5-Base (220M parameters) was also used for comparison.
  - **Training:** The T5-FiD models were fine-tuned using AdamW optimizer [19] with a weight decay of 1e-2 and an initial learning rate of 5 x 10^{-5} for 10 epochs. Linear warmup was applied for the first 5% of training steps. An effective batch size of 64 was used (achieved potentially through gradient accumulation). Training was performed on an Nvidia A100 GPU.
- **Retrieval Models:** Two types of retrieval models were used to generate the ranked lists (R_k):
  - **BM25:** A standard lexical retriever implemented using Pyserini [18].
  - **Contriever:** A dense retriever based on unsupervised contrastive learning [8]. The authors utilized the pre-trained `facebook/contriever` model from the Hugging Face Transformers library. For efficient vector search, a Faiss [5] flat index was employed.
- **Retrieval Depth (k):** Unless specified otherwise (e.g., in experiments varying k), the retriever typically provided the top $k = 50$ documents to the generator or for evaluation.

3.3 Evaluation Metrics

- **Downstream Task Metrics ($\mathcal{E}_M$ in Eq. 1):** Used to evaluate the generator's output against ground truth, both end-to-end and for single documents in eRAG. The specific metric depended on the dataset:
  - Exact Match (EM): For NQ, TriviaQA, HotpotQA.
  - Accuracy: For FEVER.
  - F1 Score: For WoW.
- **eRAG Aggregation Metrics ($\mathcal{E}_R$):** Used to aggregate the document-level eRAG scores into a list-level score. The paper experimented with:
  - Precision (P@k)
  - Recall (R@k)
  - Mean Average Precision (MAP)
  - Mean Reciprocal Rank (MRR) [2]
  - Normalized Discounted Cumulative Gain (NDCG@k) [11]
  - Hit Rate@k / Success@k

  Note: Special handling was mentioned for non-binary relevance scores in Precision and Hit Rate calculations (Precision uses average score, Hit Rate uses max score).
- **Correlation Metrics:** Used to measure the agreement between a retrieval evaluation method (like eRAG or a baseline) and the final end-to-end downstream task performance.
  - Kendall's Tau (τ)
  - Spearman's Rank Correlation Coefficient (ρ)

3.4 Baselines Compared Against

The eRAG method's correlation with end-to-end performance was compared against several baseline retrieval evaluation approaches:

- **Containing the Answer:** A common weak supervision method where a document is considered relevant if it contains the ground truth answer string [9, 14, 23, 26].
- **KILT Provenance:** Using the document-level relevance labels provided within the KILT benchmark itself (based on human annotation indicating which passages support the answer).
- **Annotation with LLM (LLM-as-Judge):** Using an external LLM (specifically, `Mistral-7B -Instruct-v0.2` [12]) to assess the relevance of each document to the query, independent of the downstream task performance. This functions similarly to methods proposed in [6, 22] but applied using a different LLM than the generator.

4 Reproducibility Implementation Details

This section details the specific steps, tools, and configurations used in our attempt to reproduce the experiments described in the eRAG paper [27]. As the original authors primarily provided the `erag.eval()` function, the RAG pipeline components—including data preprocessing, retrieval model indexing and querying, and the FiD-based generator fine-tuning and inference—were implemented from scratch based on the descriptions in the paper and common practices.

4.1 Development Environment

- **Platform:** Google Colab (Free Tier)
- **Hardware:** Nvidia T4 GPU with approximately 15GB of VRAM. This represents a significant constraint compared to the Nvidia A100 GPU used in the original study.
- **Software:** Python 3, leveraging core libraries including PyTorch, Hugging Face Transformers, Faiss (CPU version), NLTK, `rank_bm25`, and the authors' `erag` package.

4.2 Data Acquisition and Preprocessing

- **Datasets:** The initial focus for reproducibility was the Natural Questions (NQ) dataset from the KILT benchmark [20]. For the preliminary experiment detailed in Section 5, only the development set (`nq-dev-kilt.jsonl`) was downloaded and utilized. This development set was then subsequently split into training and testing subsets (*Note: This splitting of the development set represents a deviation from standard practice, undertaken for efficiency in the initial run*). Gold answers were extracted from the `output` field of the JSONL records within the development set.
- **KILT Wikipedia Corpus:**
  - **Source:** The KILT knowledge source (`kilt_knowledgesource.json`) was accessed directly via its public URL provided by the KILT authors.
  - **Preprocessing:** The corpus was processed line-by-line using custom Python scripts (`process _kilt_page` function in the provided notebook). Following the paper's description [27] and Karpukhin et al. [14], each article's text content was split into passages of a maximum of 100 words (`split_into_passages` function). Each resulting passage was then concatenated with its corresponding article title using a `[SEP]` separator (e.g., `"Article Title [SEP] Passage text..."`) to form the final documents for retrieval.

  - **Subsetting Limitation:** Due to processing time and memory constraints within the Colab environment for indexing the full multi-gigabyte corpus, initial experiments were conducted using only a small fraction (approximately 1/1000, specifically the first 5903 records) of the full KILT knowledge source. The processed passages from this subset were saved to `wikipedia_passages_sample.jsonl`. This subsetting is a major deviation from the original paper and significantly impacts the retriever's ability to find relevant information.

4.3 Retrieval Component Implementation

Two retrieval methods were implemented, mirroring the original paper:

4.3.1 BM25 Retriever

- **Libraries:** `rank_bm25` package for the BM25 algorithm and `nltk` for text preprocessing.
- **Preprocessing:** Implemented via the `preprocess_text` function. Steps included: lowercasing, punctuation removal, tokenization (`nltk.word_tokenize`), English stopword removal (`nltk.corpus.stopwords`), and Porter stemming (`nltk.PorterStemmer`). Only alphanumeric tokens were retained.
- **Indexing & Retrieval:** The `BM25Okapi` object was initialized with the tokenized documents from the subset corpus. The `bm25_retrieve` function performed retrieval by preprocessing the query using the *same* steps and then calling the `get_scores` method of the index.

4.3.2 Contriever (Dense Retriever)

- **Libraries:** Hugging Face `transformers` for the model/tokenizer and `faiss-cpu` for efficient vector search.
- **Model:** The pre-trained `facebook/contriever` model checkpoint was loaded using `AutoModel` and `AutoTokenizer`.
- **Document Encoding:** Implemented via the `encode_documents` function. Documents were processed in batches, tokenized (padding and truncation applied), and fed through the Contriever model on the available device (`cuda` if possible). Embeddings were generated via mean pooling over the last hidden states, weighted by the attention mask. Crucially, embeddings were L2 normalized as required for cosine similarity / inner product search.
- **Faiss Indexing:** A Faiss index using `IndexFlatIP` (Inner Product) was built. Document embeddings were explicitly normalized again (`faiss.normalize_L2`) before being added to the index (`index.add`).
- **Retrieval:** Implemented via the `dense_retrieve` function. Queries were encoded and L2 normalized using the same procedure as documents. The `index.search` method was used to find the nearest neighbors based on inner product similarity.
- *Note:* While implemented, Contriever was not used in the *initial preliminary experiment* reported due to time and focus constraints. Specifically, the indexing was completed, but evaluation runs were deferred.

4.4 Generation Component Implementation (T5-Small + FiD)

4.4.1 Model

The `t5-small` model (60M parameters) was loaded using `T5ForConditionalGeneration` from the `transformers` library.

4.4.2 Fusion-in-Decoder (FiD) Implementation

The FiD architecture [10] was implemented for both fine-tuning and inference:

- **Encoding:** For a given query and N retrieved documents, N distinct input sequences were created, each consisting of `"question: {query} context: {document}"`. These N sequences were tokenized and passed *independently* through the T5 encoder.
- **Decoder Input:** The resulting N sets of encoder hidden states were concatenated along the sequence length dimension. Similarly, their corresponding attention masks were concatenated to form the cross-attention mask for the decoder.
- **Generation:** The T5 decoder received the concatenated hidden states and the combined cross-attention mask to generate the final output sequence. This was handled by the `model(...)` call during training and `model.generate(...)` during inference (using the `T5_text_generator` function).

4.4.3 Fine-tuning Pipeline

- **Dataset:** An augmented NQ dataset (`augmented_nq_dataset.json`) was created using the queries and gold answers derived from the NQ development set (`nq-dev-kilt.jsonl`). Each sample contained the query, the first listed gold answer from the KILT record, and the list of 50 documents retrieved by BM25 for that query from the subset corpus. A custom PyTorch `Dataset` class (`QA_Dataset_FiD`) was implemented to handle this structure.
- **Collation:** A custom collate function (`collate_fn_fid`) was designed to handle the FiD structure. It takes batches of queries (each with a list of document encodings) and pads each document's tokenized input (`input_ids`, `attention_mask`) to `max_input_length`. It also pads the list of documents per query up to a fixed `max_docs_per_item` (set to 40 in the preliminary run) for consistent tensor shapes. Target labels (`gold_answer`) were padded to `max_target_length`.
- **Training Setup:** The optimizer was AdamW (lr=5e-5, weight_decay=1e-2). An effective batch size of 64 was simulated using gradient accumulation (`gradient_accumulation_steps = 64`) with a per-device batch size of 1 on the T4 GPU.
- **Epochs Limitation:** Due to severe time constraints on Colab T4, the preliminary fine-tuning was run for only **1 epoch**, a significant reduction from the 10 epochs reported in the original paper.
- **Linear Warmup:** Implemented for the first 5% of optimizer steps, as described in the paper.
- **Output:** The fine-tuned model was saved locally to the `finetuned_t5_model_fid` directory.

4.5 Evaluation Pipeline Implementation

- **eRAG Evaluation:** The core evaluation logic was performed using the `erag.eval` function provided by the original authors. This function orchestrates the per-document generation and downstream metric calculation.
- **Downstream Metric:** A custom `exact_match_metric` function was implemented. It normalizes both the generated output and the gold standard answers by lowercasing, removing punctuation (`string.punctuation`), and stripping extra whitespace (`normalize_answer` function) before checking for an exact match against any of the provided gold answers for a query.
- **End-to-End Evaluation:** Performed by calling the fine-tuned T5-FiD generator (`t5_generator_for_eval`) with the full list of k retrieved documents for each query in the test set (`test_retrieval_results`). The resulting dictionary of generated answers was then evaluated using the same `exact_match_metric`.
- **Correlation:** Spearman's Rho and Kendall's Tau correlations between the aggregated eRAG scores (e.g., `P_40`, `success_40` from `erag_results['per_input']`) and the end-to-end scores (`e2e_scores_dict`) were calculated using `scipy.stats.spearmanr` and `scipy.stats.kendalltau`.

- **Logging:** Per-query eRAG results, aggregated eRAG metrics, and end-to-end scores were saved to separate JSON files within a `logs` directory for inspection. A checkpoint file (`experiment_checkpoint.pkl`) was used to save intermediate correlation results.

5 Preliminary Results and Discussion

This section presents the results obtained from an initial, limited-scale experiment conducted as part of this reproducibility effort. It compares these findings to those reported in the original eRAG paper and discusses potential reasons for observed discrepancies, largely stemming from resource constraints.

5.1 Experiment Configuration

The preliminary experiment was configured as follows:

- **Dataset:** Natural Questions (NQ) development set (used as the test set for evaluation after splitting off a training portion).
- **Retriever:** BM25 (implemented using `rank_bm25` and `nltk`).
- **Corpus:** A small subset of the KILT Wikipedia corpus, comprising approximately 1/1000th of the full data (first 5900 records processed into 216k passages).
- **Generator:** T5-Small with Fusion-in-Decoder (FiD), fine-tuned for **1 epoch** on the NQ training split using the setup described in Section 4.4.3.
- **Retrieval Depth:** $k = 40$ documents were retrieved by BM25 for each query.
- **Evaluation:**
  - Downstream Metric: Exact Match (EM), normalized.
  - eRAG Aggregation Metrics: Precision@40 (`P_40`) and Success@40 (`success_40`).
  - Correlation Metrics: Spearman's Rho and Kendall's Tau, comparing eRAG scores (`P_40`, `success_40`) against end-to-end EM scores.

5.2 Observed Correlations

The correlation analysis between the eRAG scores and the end-to-end performance (Exact Match) on the NQ test split yielded the following results:

- **For eRAG metric `P_40` (BM25, K=40):**
  - Spearman correlation (ρ): 0.234 (p $\approx$ 0.000)
  - Kendall correlation (τ): 0.232 (p $\approx$ 0.000)
- **For eRAG metric `success_40` (BM25, K=40):**
  - Spearman correlation (ρ): 0.234 (p $\approx$ 0.000)
  - Kendall correlation (τ): 0.234 (p $\approx$ 0.000)

These results indicate a statistically significant, positive correlation between the eRAG metrics calculated using our pipeline and the final end-to-end task performance, even under these limited conditions. Both Precision@40 and Success@40 showed similar levels of correlation. **It is important to note, however, that these correlations were observed despite extremely low absolute performance levels of the system in this preliminary run (detailed further in Section 5.4).**

5.3 Comparison with Paper's Claims

While the observed correlations are positive and significant, they are notably lower than those reported by Salemi & Zamani [27]. In Table 1 of their paper, using BM25 on the NQ dataset with $k = 50$, they report Kendall's Tau correlations around:

- ∼0.53 for P@50
- ∼0.52 for Hit Rate@50 (conceptually similar to Success@k)

Our preliminary Kendall Tau values of approximately 0.23 are substantially lower than the ∼0.53 reported for Precision and the ∼0.52 for Hit Rate/Success.

5.4 Analysis of Discrepancies and Challenges

Several factors, primarily stemming from the significant difference in available computational resources compared to the original study, likely contribute to these discrepancies:

1. **Extremely Low End-to-End Performance:** Examination of the generated outputs and evaluation logs (`logs/end_to_end_BM25_K40.json`) revealed that the fine-tuned T5-Small model achieved only 6 correct answers out of 568 test queries, resulting in an end-to-end Exact Match score of approximately **1.06%**. This extremely low score indicates the generator was largely unable to perform the task correctly under these conditions. Such low performance, with very little variance in the outcome, makes it statistically challenging to establish a strong correlation with any other metric, as there's little successful performance to differentiate.

2. **Hardware Constraints and Fine-tuning:** The original study utilized an Nvidia A100 GPU, whereas this reproduction was limited to a Colab T4 GPU. This difference impacts training stability, achievable batch sizes (necessitating greater gradient accumulation), and overall training time. Fine-tuning large models effectively often requires significant computational power.

3. **Insufficient Fine-tuning:** The model was fine-tuned for only **1 epoch**, compared to the 10 epochs used in the original paper. It is highly probable that the T5-Small model did not converge adequately within a single epoch, directly contributing to its poor downstream performance (1.06% EM). Since eRAG relies on this same generator to evaluate document utility, the generator's inability to effectively use evidence significantly impacts the validity and usefulness of the resulting eRAG scores (P@40 and Success@40) themselves.

4. **Limited Corpus Size and Poor Document Utility:** Using only ∼1/1000th of the Wikipedia corpus severely restricted the BM25 retriever's ability to find genuinely relevant and informative passages. The extremely low aggregated eRAG scores quantitatively reflect this: the observed **Precision@40 was only 0.0047**, meaning that, on average, the documents retrieved and assessed by the poorly trained T5 model were deemed almost entirely non-useful. Furthermore, the **Success@40 was only 0.074**, indicating that for over 92% of queries, **none** of the top 40 documents retrieved from the subset corpus allowed the generator to produce a correct response according to the EM metric. When the input documents themselves lack utility (due to poor retrieval from a sparse corpus), it further hampers the ability to observe meaningful correlations.

5. **Data Strategy Deviation for Efficiency:** As noted in the implementation details, to expedite the preliminary experiments within the available time, the fine-tuning was performed on a split of the *development* set, with the remaining portion used for testing. This deviates from the standard protocol of fine-tuning on the designated KILT *training* set (`nq-train-kilt.jsonl`) and evaluating on the development set (`nq-dev-kilt.jsonl`). Correcting this by using the official KILT training data is a necessary step for future work.

6. **Preprocessing Nuances:** While efforts were made to follow the described preprocessing, minor differences or edge cases (like handling semi-empty passages resulting from the 100-word split) could potentially introduce noise.

In summary, the preliminary results show a positive correlation trend as claimed by the eRAG paper, but the magnitude is significantly reduced. This is strongly suspected to be due to the compounding effects of limited computational resources, insufficient model fine-tuning (leading to near-zero generator performance), and inadequate corpus coverage severely limiting retriever effectiveness in this initial reproducibility attempt.

6 Challenges Encountered and Future Work

Attempting to reproduce the results of the eRAG paper presented several significant challenges, primarily related to computational resources and the complexity of reconstructing the full experimental pipeline. This section summarizes these hurdles and outlines the necessary next steps for a more comprehensive reproducibility study.

6.1 Summary of Challenges

1. **Hardware Limitations:** The most significant bottleneck was the difference in computational hardware. The original study utilized Nvidia A100 GPUs, whereas this project was constrained to a Google Colab T4 GPU with limited VRAM (approx. 15GB). This disparity severely impacted:
   - **Fine-tuning Duration:** Running the T5-Small-FiD fine-tuning for the reported 10 epochs was infeasible within Colab time limits. The preliminary run was restricted to only 1 epoch, which was clearly insufficient for model convergence, leading to poor generator performance.
   - **Model Scale:** While T5-Small was manageable, attempting to reproduce experiments with T5-Base (used for comparison in the original paper) would likely be impossible on the T4 GPU.
   - **Batch Sizes & Stability:** Smaller per-device batch sizes and increased gradient accumulation were necessary, potentially affecting training dynamics compared to the original setup.
2. **Pipeline Reconstruction Effort:** The authors provided the core `erag.eval` function, but the surrounding RAG architecture—including the specific implementations of BM25 retrieval, Contriever indexing and retrieval, data loading/preprocessing for KILT, and particularly the T5-FiD fine-tuning and inference logic—had to be built from scratch based on the paper's descriptions and related works [10]. This required significant implementation effort and introduced potential minor variations from the original authors' internal code.
3. **Corpus Scale and Indexing:** The KILT Wikipedia dump is a large corpus (tens of gigabytes). Processing, tokenizing, and indexing this entire corpus for both BM25 (requiring significant RAM) and Contriever (requiring large embedding storage and Faiss index construction) proved challenging within the Colab environment. The necessary workaround of using a tiny (1/1000) subset drastically limited the effectiveness of the retrieval component in the preliminary experiments.
4. **Dense Retrieval Setup:** While the Contriever model and Faiss indexing were implemented, integrating them fully into the time-constrained workflow and running evaluations requires further optimization and dedicated runs.

6.2 Planned Next Steps / Future Work

To overcome these challenges and achieve a more faithful reproduction of the eRAG paper's findings, the following steps are necessary:

1. **Secure Adequate Computational Resources:** Ideally, gaining access to GPUs comparable to the A100 used in the original study is *paramount* for effective fine-tuning and handling the full dataset scales. Alternatively, exploring more distributed or longer-running cloud computing options might be necessary.
2. **Process and Index the Full Corpus:** Implement a robust, potentially offline, pipeline to preprocess the entire KILT Wikipedia dump and build complete BM25 and Faiss (for Contriever) indices. This is crucial for evaluating the retrievers under realistic conditions.

3. **Complete T5-FiD Fine-tuning on Official Training Data:** Execute the fine-tuning process using the official KILT NQ *training set* (`nq-train-kilt.jsonl`) for the full 10 epochs specified in the paper. This corrects the data strategy deviation used in the preliminary run.

4. **Run Contriever Experiments:** Conduct the full eRAG evaluation using the implemented Contriever retriever, comparing its correlation results against BM25 and the original paper's findings (Table 1).

5. **Vary Retrieval Depth (k):** Reproduce the experiments shown in Figure 1 of the paper, analyzing how the correlation between eRAG metrics and end-to-end performance changes as k (the number of documents passed to the generator/evaluated) varies.

6. **Analyze Computational Efficiency:** Once the core functionality is reliably reproduced, systematically measure and compare the inference time and GPU memory consumption of the eRAG evaluation method (per-document generation) against the standard end-to-end evaluation (concatenated document generation), aiming to verify the claims made in Table 2 of the paper.

7. **Expand Dataset Coverage:** If resources permit, extend the reproducibility study to other KILT datasets evaluated in the original paper, such as FEVER (classification) and WoW (dialogue), to assess the generalizability of the eRAG findings across different tasks and metrics (Accuracy, F1).

Addressing these points, particularly the corpus scale and fine-tuning duration, is essential to rigorously validate the claims made about the eRAG evaluation method.

7 Conclusion

This report detailed the ongoing effort to reproduce the experimental findings of the paper "Evaluating Retrieval Quality in Retrieval-Augmented Generation" by Salemi and Zamani [cite original paper], which introduced the eRAG evaluation method. The primary goal was to reconstruct the necessary Retrieval-Augmented Generation (RAG) pipeline and validate the paper's central claim that eRAG scores correlate more strongly with downstream task performance than traditional retrieval evaluation baselines.

Significant progress was made in implementing the core components described in the paper from scratch. This included setting up data preprocessing for the KILT NQ dataset and Wikipedia corpus, implementing both BM25 and Contriever retrieval systems (with Faiss indexing for the latter), successfully fine-tuning a T5-Small model using the Fusion-in-Decoder (FiD) architecture, and integrating the authors' `erag.eval` function with a custom Exact Match metric for evaluation.

Preliminary experiments, conducted under significant resource constraints (using a T4 GPU, a 1/1000 corpus subset, and only 1 epoch of fine-tuning), demonstrated a statistically significant positive correlation (Kendall's Tau ≈ 0.23) between eRAG metrics (P@40, Success@40) and end-to-end performance on the NQ dataset using BM25. However, this correlation was substantially lower than reported in the original study, occurring alongside extremely low absolute system performance (**approximately 1.06% end-to-end Exact Match**). The discrepancy is strongly attributed to the limitations faced, particularly the insufficient fine-tuning of the T5-FiD generator and the sparse coverage of the retrieval corpus subset, which likely led to poor overall RAG performance and consequently weakened the observed correlation signal.

This reproducibility study remains a work in progress. Key future work involves securing adequate computational resources to process the full Wikipedia corpus, complete the full 10 epochs of T5-FiD fine-tuning, conduct experiments with the Contriever retriever, extend the study to cover all the datasets in the original paper, and perform the computational efficiency comparisons. Successfully completing these steps is essential for a comprehensive validation of the eRAG method.

Despite the preliminary limitations, the successful pipeline reconstruction and the observation of a positive correlation trend provide a foundation for future work. If the higher correlation and efficiency claims of eRAG hold under more rigorous reproduction, it represents a valuable contribution to the challenging but crucial task of evaluating and optimizing retrieval components within complex RAG systems.

References

[1] Garima Agrawal, Tharindu Kumarage, Zeyad Alghamdi, and Huan Liu. Can knowledge graphs reduce hallucinations in llms?: A survey. *arXiv preprint arXiv:2311.07914*, 2023.

[2] Nick Craswell. *Mean Reciprocal Rank*, pages 1703–1703. Springer US, Boston, MA, 2009.

[3] Romain Deffayet, Philipp Hager, Jean-Michel Renders, and Maarten de Rijke. An offline metric for the debiasedness of click models. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 558–568, 2023.

[4] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*, 2018.

[5] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library, 2025.

[6] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta, March 2024. Association for Computational Linguistics.

[7] Fan Guo, Chao Liu, and Yi Min Wang. Efficient multiple-click models in web search. In *Proceedings of the second acm international conference on web search and data mining*, pages 124–131, 2009.

[8] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.

[9] Gautier Izacard and Edouard Grave. Distilling knowledge from reader to retriever for question answering. *arXiv preprint arXiv:2012.04584*, 2020.

[10] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*, 2020.

[11] Kalervo Järvelin and Jaana Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, volume 51, pages 243–250. ACM New York, NY, USA, 2017.

[12] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[13] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

[14] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781, 2020.

[15] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[16] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[17] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[18] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2356–2362, 2021.

[19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[20] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. Kilt: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*, 2020.

[21] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[22] Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. Ares: An automated evaluation framework for retrieval-augmented generation systems, 2024.

[23] Alireza Salemi, Juan Altmayer Pizzorno, and Hamed Zamani. A symmetric dual encoding dense retrieval framework for knowledge-intensive visual question answering. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 110–120, 2023.

[24] Alireza Salemi, Surya Kallumadi, and Hamed Zamani. Optimization methods for personalizing large language models through retrieval augmentation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 752–762, 2024.

[25] Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. Lamp: When large language models meet personalization. *arXiv preprint arXiv:2304.11406*, 2023.

[26] Alireza Salemi, Mahta Rafiee, and Hamed Zamani. Pre-training multi-modal dense retrievers for outside-knowledge visual question answering. In *Proceedings of the 2023 ACM SIGIR international conference on theory of information retrieval*, pages 169–176, 2023.

[27] Alireza Salemi and Hamed Zamani. Evaluating retrieval quality in retrieval-augmented generation, 2024.

[28] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*, 2021.

[29] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*, 2018.

[30] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

[31] Hamed Zamani and Michael Bendersky. Stochastic rag: End-to-end retrieval-augmented generation through expected utility maximization. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2641–2646, 2024.

[32] Hamed Zamani, Fernando Diaz, Mostafa Dehghani, Donald Metzler, and Michael Bendersky. Retrieval-enhanced machine learning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2875–2886, 2022.