RefineLoop RAG: Dynamic Query Refinement through Iterative Evaluation by LLM

Yongqi Chen, Zimo Si, Logan Ju

Abstract

This report presents a novel query optimization approach within the Retrieval-Augmented Generation (RAG) framework, focusing on iterative refinement through model-driven evaluation. The proposed method dynamically adjusts user queries based on the evaluation of initial retrieval results using generative and discriminative scoring models. By leveraging few-shot examples and binary equivalence prompts, the system ensures precise alignment between model-generated outputs and ground-truth labels. The evaluation process employs LLaMA and GPT models, using tailored prompts to determine semantic equivalence between answers. Results demonstrate the effectiveness of this iterative evaluation mechanism in enhancing query precision and retrieval relevance, contributing to the development of robust and adaptable RAG systems. This work highlights the potential of combining retrieval and evaluation loops for improving performance in knowledge-intensive tasks.

We have made our code available at http://github.com/LoganJu2000/595RAG

1 Introduction

The exponential growth of information and the increasing complexity of user queries have underscored the importance of efficient knowledge retrieval and integration in artificial intelligence systems. Retrieval-Augmented Generation (RAG) has emerged as a key paradigm, enabling language models to generate informed and contextually relevant responses by integrating external knowledge bases. By combining the strengths of retrieval systems and generative models, RAG systems provide a powerful solution for knowledge-intensive tasks, including question answering, summarization, and content synthesis. However, the effectiveness of traditional RAG systems is often constrained by their reliance on static query formulations, leading to suboptimal performance when initial retrieval attempts fail to capture the full scope of user intent.

To address these limitations, this paper proposes a novel approach called Refine Loop RAG that introduces iterative evaluation loops into the RAG framework, enabling continuous query refinement. The proposed methodology dynamically adjusts queries based on the evaluation of initial retrieval results, leveraging a hybrid scoring mechanism that combines outputs from generative and discriminative models. This evaluation-driven process ensures precise alignment between user queries, retrieved knowledge, and generated outputs, minimizing the need for manual intervention.

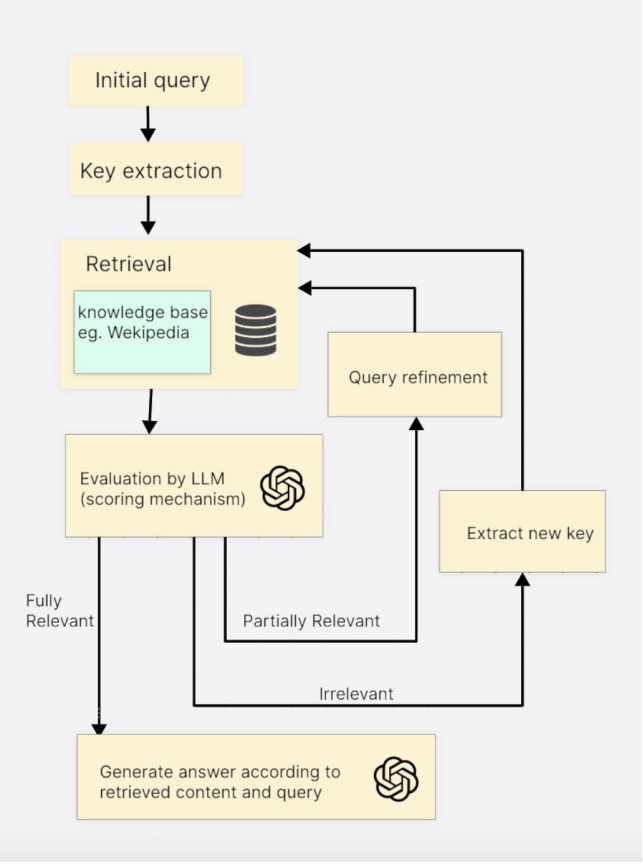

Figure 1: This is the diagram of the pipeline of query refinement through iterative evaluation

2 Related Work

Retrieval-Augmented Generation (RAG) has emerged as a transformative framework in natural language processing (NLP), enabling large language models (LLMs) to integrate external knowledge during generation tasks. This approach addresses key limitations of LLMs, such as hallucinations, outdated knowledge, and a lack of transparency in reasoning. RAG was first introduced by Lewis et al. (Lewis et al., 2020), where it was applied to knowledge-intensive NLP tasks like open-domain question answering. The framework combines a retrieval component with a generative model to enhance the accuracy and credibility of generated content.

Early implementations, often referred to as Naïve RAG, employed a simple pipeline where user queries were encoded, documents were retrieved based on semantic similarity, and the retrieved content was appended to the query for generation. While effective, these systems heavily relied on the quality of initial queries and retrieved documents, leading to limitations in precision and recall. To address these challenges, Izacard and Grave (Izacard and Grave, 2021) introduced dense retrieval and better integration strategies between retrieval and generation, setting the foundation for more advanced RAG systems.

Advanced RAG frameworks expanded on these ideas by incorporating innovations such as fine-grained chunking, metadata-enhanced retrieval, and re-ranking mechanisms. These techniques improved retrieval accuracy and reduced noise in the retrieved content. However, the sequential structure of these systems often restricted their adaptability to complex, multi-step reasoning tasks (Khattab et al., 2021).

The advent of Modular RAG systems marked a significant evolution in the field. Modular RAG decouples the retrieval, generation, and augmentation processes, allowing for dynamic configurations tailored to specific tasks. Recent works, such as those by Shi et al. (Shi et al., 2022), introduced adaptive feedback mechanisms and iterative retrieval strategies, enabling the system to refine queries and improve retrieval quality dynamically. These innovations have proven effective in addressing the limitations of earlier RAG models and enhancing their scalability and versatility.

A crucial advancement in RAG systems has been query optimization. Techniques such as query rewriting, hypothetical document generation (e.g., HyDE), and multi-query expansion have significantly improved the alignment between user queries and relevant documents. Such techniques are particularly effective in modular systems, as highlighted by Gao et al. (Gao et al., 2024), where iterative refinement through feedback loops further enhances retrieval precision and relevance.

Query transformation has emerged as a critical technique in improving the performance of retrieval-augmented systems. More recent advances have introduced dynamic and context-aware query refinement techniques. Approaches like HyDE (Hypothetical Document Embeddings) generate hypothetical documents based on user queries, which are then used to improve retrieval relevance (Izacard and Grave, 2021). Similarly, multi-query expansion frameworks explore diverse reformulations of the original query to maximize retrieval coverage and reduce ambiguity. These techniques are particularly beneficial in open-domain question-answering systems, where initial queries may lack specificity.

In the context of Retrieval-Augmented Generation (RAG), iterative query refinement loops represent a notable innovation. By integrating generative and discriminative models, these systems dynamically adjust queries through evaluation mechanisms, ensuring alignment with user intent and improving retrieval accuracy. The use of hybrid scoring, which combines semantic similarity metrics with generative evaluation, has shown to significantly improve query transformation effectiveness (Shi et al., 2022) (Zheng et al., 2024). These methods highlight the evolving importance of dynamic and iterative query transformation in building robust knowledge-intensive AI systems.

Despite these advancements, challenges remain in scaling RAG systems to large and diverse datasets, ensuring robustness against noisy data, and integrating multi-modal knowledge. Future research directions include leveraging reinforcement learning for retriever-generator alignment, developing real-time adaptive retrieval systems, and extending RAG capabilities to multi-modal and real-time applications (Gao et al., 2024).

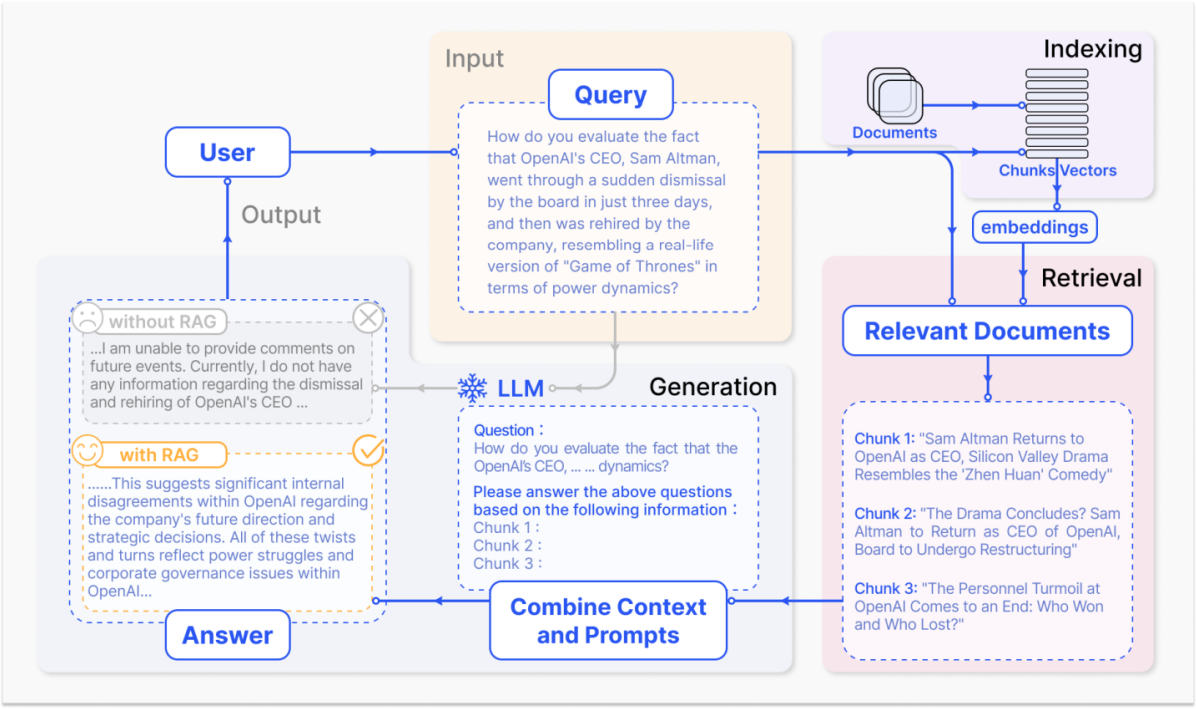

Figure 2: The Naive RAG process(Gao et al., 2024) for question answering involves three main steps: Indexing: Documents are divided into chunks, encoded into vectors, and stored in a vector database. Retrieval: The top-k chunks most relevant to the question are retrieved based on semantic similarity. Generation: The question and retrieved chunks are input into an LLM to generate the final answer.

3 Proposed Method: RefineLoop RAG

3.1 Naïve Retrieval-Augmented Generation (RAG)

We will use Naïve Retrieval-Augmented Generation (RAG) as the baseline system. It is a framework designed to enhance language models by integrating external knowledge bases during generation tasks. In its naïve form, the process involves three primary steps:

1. **Query and Retrieval:** A user-provided query is submitted to a retrieval system, which fetches relevant documents from a predefined knowledge base based on similarity metrics, such as dense embeddings.

2. **Context Injection:** The retrieved documents are concatenated with the query to form a context-enriched input, which is then passed to a generative model.

3. **Generation:** The generative model produces an output, such as an answer or summary, conditioned on the query and retrieved context.

While effective, the naïve RAG approach relies heavily on the quality of the initial query and retrieved documents. It lacks mechanisms for iterative refinement, often leading to suboptimal results when the initial retrieval is incomplete or misaligned.

3.2 Core Idea

To address the limitations of naïve RAG, our method introduces iterative evaluation loops to dynamically refine user queries. By using the initial retrieval results as evaluation, the approach continuously optimizes query formulation and improves retrieval accuracy. This reduces reliance on user intervention and ensures a closer alignment between the query and relevant information.

3.3 Methodology

3.3.1 Initial Query and Retrieval

- The process begins with an initial user query, which is submitted to the retrieval module.

- The retrieval module searches the knowledge base using semantic embeddings or other similarity measures to fetch a ranked list of relevant documents.

- The retrieved documents serve as the basis for subsequent refinement and evaluation.

3.3.2 Evaluation

- The retrieved documents are evaluated by a scoring system that combines outputs from a generative model and a discriminative model.
- Each document is categorized into three relevance levels: *irrelevant*, *partially relevant*, or *fully relevant*, based on its alignment with the query.
- This evaluation provides actionable evaluation for improving the query.

3.3.3 Query Refinement

- The query is adjusted based on the evaluation by modifying keywords, rephrasing semantic structures, or shifting focus to more relevant aspects.
- These adjustments are guided by prompt engineering techniques designed to extract more precise or disambiguated entities from the user query by the generative model.

3.3.4 Iterative Evaluation Loop

- The refined query is resubmitted to the retrieval system for a new round of document retrieval.
- This iterative process is repeated until the retrieval results meet a predefined relevance threshold or reach convergence, ensuring optimal alignment between the query and the desired information.

3.4 Implementation Details

The proposed iterative query improvement framework is implemented with the following steps:

1. **Initial Query Processing:** Each query from the dataset is analyzed to extract key terms using a title extraction algorithm. If no relevant terms are found, the query is skipped.
2. **External Knowledge Retrieval:** Based on the extracted terms, the system retrieves relevant content from Wikipedia using the API. The retrieved content is processed to remove irrelevant sections and embedded into a semantic vector space using a text embedding model.
3. **Iterative evaluation Loop:** For each query, the retrieved content is evaluated using a evaluation mechanism:
   - If the content is *fully relevant*, the query and retrieved content are passed to a generative model (e.g., LLaMA) to produce the final answer.
   - If the content is *partially relevant*, the query is refined using a query adjustment function based on evaluation and resubmitted for retrieval.
   - If the content is *irrelevant*, new key terms are extracted, and the retrieval process is repeated.
4. **Answer Generation:** Once the evaluation loop converges or meets the predefined threshold, the optimized query and relevant content are passed to the generative model to produce the final answer.

3.5 Key Innovations

- **Iterative Optimization:** The method integrates an iterative evaluation loop into the RAG pipeline, addressing the inherent limitations of one-pass retrieval.
- **Hybrid Scoring System:** By combining generative and discriminative models for document evaluation, the approach achieves a more nuanced assessment of relevance.
- **Enhanced Prompt Engineering:** The query refinement process leverages advanced prompt design to dynamically improve the quality and specificity of queries across iterations.
- **Reduced User Burden:** The system minimizes manual query adjustments by automating the evaluation and refinement process.

4 Evaluation Plan

To evaluate the performance of our query optimization method within the Retrieval-Augmented Generation (RAG) framework, we adopt a few-shot evaluation approach inspired by (Zheng et al., 2024). This approach focuses on assessing the semantic equivalence between model-generated outputs and the ground-truth target labels.

4.1 Few-Shot Evaluation Setup

The evaluation process involves providing the scoring model with structured prompts and examples to guide it in classifying the equivalence of answers. The main steps are:

Algorithm 1 Iterative Query Refinement for RefineLoop RAG

Input: Initial query q, Maximum iterations T, evaluation thresholds.
Output: Final answer A

1. Extract key terms from the initial query q using title extraction 2. Retrieve relevant Wikipedia content using extracted terms 3. Embed retrieved content into semantic vector space using embedding model 4. Initialize $current\_query \leftarrow q$ and $iteration \leftarrow 1$

while $iteration \leq T$ **do**

5. Query the vector search engine using $current\_query$ to fetch top-ranked results 6. Evaluate the retrieved content using evaluation mechanism: **if** *evaluation is fully relevant* **then**
   7. Generate answer A using LLaMA with query q and retrieved content **return** A
   else
   if *evaluation is partially relevant* **then**
   8. Refine $current\_query$ using evaluation and re-query
   else
   9. Extract new key terms from $current\_query$ and retrieve fresh Wikipedia content
10. Increment $iteration$ by 1

11. Generate final answer A using latest retrieved content and q **return** A

- **Prompt Design:** Each prompt includes a question, two answers (one generated by the model and one ground truth), and an instruction to decide whether the two answers are equivalent based solely on their semantic meaning. The model outputs a binary decision: `"Yes"` for equivalent and `"No"` for non-equivalent.

- **Few-Shot Examples:** To guide the scoring model, a few-shot setup is included in the prompt. Positive examples demonstrate pairs of equivalent answers, while negative examples highlight non-equivalent cases. For instance:

  > **Question:** Which title was conferred to Anna Muzychuk in 2007?
  > **Answer 1:** Anna Muzychuk was conferred the title of International Master (IM) in 2007.
  > **Answer 2:** International Master.
  > **Are the two answers equivalent?**
  > `"Yes"`

  Such examples help the model understand the equivalence criteria.

4.2 Evaluation Process

The Llama-3-8B and GPT4o-mini models are evaluated based on their ability to generate outputs that align with ground-truth labels in terms of semantic equivalence. The scoring model parses the outputs as `"Yes"` or `"No"` to indicate equivalence or non-equivalence.

4.3 Results Parsing

The evaluation framework directly interprets the scoring model's binary outputs (`TRUE` or `FALSE`) to determine if the model-generated answer is equivalent to the target label. This provides a straightforward measure of success for individual test cases without relying on aggregate metrics.

This evaluation plan emphasizes semantic alignment between model outputs and target labels through iterative prompts and a robust few-shot mechanism. By focusing on binary equivalence judgments, the evaluation ensures clarity and reliability in determining the effectiveness of the query optimization approach.

4.4 Additional: Similarity-Based Evaluation

To further evaluate the performance of the query optimization method, we leverage a similarity-based approach using the Hugging Face sentence-transformers model `all-MiniLM-L6-v2`. This method assesses the semantic similarity between model-generated answers and target answers, providing a quantitative measure of alignment.

The evaluation involves the following steps:

1. **Sentence Embedding:** Both the model-generated answer (`Answer1`) and the target answer (`Answer2`) are encoded into semantic embeddings using the Hugging Face API.

2. **Similarity Calculation:** The embeddings are compared to compute a similarity score, ranging from 0 to 1, where 1 indicates perfect semantic equivalence.

Question	When did Richmond last play in a preliminary final?
Answer 1	Richmond last played in a preliminary final in 2020.
Answer 2	Richmond's most recent preliminary final appearance was in 2020.
Answer 1 (short)	2020
Answer 2 (short)	2020
Are the two answers equivalent?	`Yes`
Question	When did Richmond last play in a preliminary final?
Answer 1	Richmond last played in 2019.
Answer 2	Richmond's most recent preliminary final was in 2020.
Answer 1 (short)	2019
Answer 2 (short)	2020
Are the two answers equivalent?	`No`
Question	`<Question>`
Answer 1	`<Model Output>`
Answer 2	`<Target Label>`

Table 1: Illustration of few-shot evaluation with the model

3. **Scoring:** A threshold (e.g., 0.8) is used to classify the answers as equivalent or non-equivalent. Scores above the threshold indicate strong semantic alignment.

The similarity calculation is performed using the Hugging Face inference API. The Python function `calculate_similarity` sends the answers to the API, which returns the similarity score based on pre-trained embeddings. The process is robust to minor lexical differences while capturing semantic relationships.

For demonstration, consider the following input:

- **Answer 1:** `"That is a happy person"`
- **Answer 2:** `"That is a very happy person"`

The calculated similarity score is `0.95`, indicating strong semantic equivalence between the two answers.

This evaluation method provides a quantitative and automated approach to assess semantic alignment, complementing binary equivalence judgments. However, it is sensitive to context mismatches and may not fully capture domain-specific nuances.

By integrating this similarity-based evaluation, we enhance the robustness of the assessment framework, ensuring a comprehensive evaluation of the query optimization method.

5 Experiment

5.1 Dataset

In the experiment, we use the Natural Questions (NQ) dataset, sourced from **Hugging Face**, is designed for open-domain question answering (QA), a benchmark task in artificial intelligence. It evaluates the ability of systems to read and comprehend text to answer complex questions, bridging the gap between research and real-world applications in information retrieval.

Key Features:

- **Real User Questions:** The dataset consists of questions posed by real users, offering a realistic foundation for training and evaluation.
- **Full Wikipedia Articles:** QA systems must process entire Wikipedia articles to find answers, simulating real-world conditions.
- **Challenge Level:** The inclusion of natural user queries and unstructured text makes this dataset more challenging compared to previous QA datasets.

The dataset supports research in natural language processing, text comprehension, and information retrieval. It is a valuable resource for developing systems that mimic human-like understanding and reasoning capabilities.

5.2 External Knowledge Base

The Wikipedia dataset, hosted by Hugging Face, serves as the foundational knowledge base for our RAG (Retrieval-Augmented Generation) system. This dataset provides a curated collection of articles from Wikipedia, encompassing content in multiple languages, and is widely used for various natural language processing tasks such as language modeling, information retrieval, and text generation.

Each entry in the dataset includes the following fields:

- **id**: A unique identifier for the article.
- **url**: The URL pointing to the original Wikipedia page.
- **title**: The title of the article.
- **text**: The full text content of the article, cleaned and stripped of unnecessary elements such as references, markdown, and metadata.

The dataset is constructed using dumps from the official Wikipedia **repository**and is pre-processed with the `mwparserfromhell` library. This tool ensures that the data is cleaned, normalized, and prepared for downstream applications by removing formatting inconsistencies and unwanted sections.

A notable feature of the dataset is its comprehensive multi-language support, encompassing over 291 languages. Each language split corresponds to a specific dump and is stored separately, making the dataset highly versatile for cross-lingual and multilingual use cases. Pre-processed subsets are readily available for popular languages such as English, French, German, and Italian, with configurations like `"20220301.en"` and `"20220301.fr"` being directly loadable using Hugging Face's `datasets` library.

The dataset supports efficient usage in large-scale projects through features like parallel data generation (`num_proc`) and flexibility in selecting specific language subsets or date versions. These features enable the dynamic construction of a knowledge base tailored to specific research or application requirements.

This dataset is co-licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License (CC BY-SA) and the GNU Free Documentation License (GFDL), ensuring that it is openly accessible for research and commercial applications while adhering to proper licensing practices. Its versatility, scale, and multi-language coverage make it an ideal resource for powering retrieval-augmented systems requiring extensive and diverse knowledge bases.

To complement the static Wikipedia dataset, we implement a dynamic retrieval mechanism that allows the system to fetch the latest content directly from Wikipedia based on user queries. This ensures that the knowledge base remains up-to-date and provides relevant context for the Retrieval-Augmented Generation (RAG) process.

5.2.1 Dynamic Retrieval Process

The retrieval process starts with the user-provided query, which is sent to the Wikipedia API for searching relevant pages. The top-ranked page is selected based on its semantic alignment with the query. The system then fetches the full content of the identified page, which is parsed and cleaned to extract the main textual content, such as introductory paragraphs and key sections.

5.2.2 Content Cleaning and Structuring

After retrieving the raw HTML content of a Wikipedia page, it is processed to remove unnecessary metadata, such as references, formatting tags, and unrelated sections. This cleaning step ensures that the extracted content is concise and focused, suitable for feeding into the RAG pipeline as input context. The final output includes:

- **Page Title:** The title of the Wikipedia page.
- **Page URL:** A direct link to the Wikipedia page for reference.
- **Page Content:** A clean, plain-text version of the page's main content, focusing on relevant paragraphs.

5.2.3 Integration with RAG

The retrieved content is then seamlessly integrated into the RAG pipeline. Once the query and its corresponding retrieved content are prepared, they are concatenated and provided as input to the generative model. This allows the model to leverage both the original query and the retrieved knowledge base for generating informed and context-aware responses.

5.3 Baseline

We built a Naïve RAG baseline from scratch. The baseline implementation of Retrieval-Augmented

Generation (RAG) combines generative and retrieval-based approaches to produce answers grounded in external knowledge. This pipeline leverages OpenAI's GPT models for query understanding, a vector-based embedding system for document retrieval, and generative models including Llama-3-8B and GPT-4o-mini for answer generation.

Our naive RAG pipeline follows these steps:

- **Keyword Extraction:** Using OpenAI's `gpt-4o-mini`, the system extracts key words or a relevant Wikipedia article title based on the user query. A carefully designed prompt ensures the extraction of concise and relevant terms.
- **Document Retrieval:** The extracted keywords are used to fetch Wikipedia article. The content is split into overlapping chunks, embedded into a semantic vector space using the Hugging Face model `BAAI/bge-small-en-v1.5`, and stored in a vector index.
- **Naïve Answer Generation:** The query is passed directly to the generative model to produce an answer without external context.
- **Naïve RAG Answer Generation:** The query and retrieved document chunks are passed to the generative model e.g. Llama-3-8B and GPT4o-mini, enabling it to generate a context-enriched answer.

The key components are

- **Generative Model:** We use Llama-3-8B capable of producing natural language answers from a given query.
- **Embedding:** `BAAI/bge-small-en-v1.5` is used for embedding Wikipedia content into a vector space. This lightweight model balances computational efficiency and accuracy.
- **Vector Indexing:** The LlamaIndex library handles text chunking, embedding, and retrieval, enabling efficient similarity-based document search.

The Parameters and Configurations are

- **Chunk Size:** Text is divided into chunks of 100 characters with an overlap of 10 characters to preserve contextual continuity.
- **Top-K Retrieval:** The query engine retrieves the top 10 most similar chunks for each query.

5.4 Iterative Evaluation Setup

The evaluations were conducted with a set of carefully chosen hyperparameters to ensure a balance between computational efficiency and performance. Below are the key hyperparameter configurations:

- **Maximum Iterations:** The iterative refinement process was limited to 3 iterations to control the computational cost while allowing sufficient opportunity for query optimization.
- **Feedback Mechanism:** The relevance of retrieved content was evaluated using `gpt-4o-mini`, which categorized content as *fully relevant*, *partially relevant*, or *irrelevant*.

The configurations ensured consistency across different experimental setups, enabling a fair comparison between models and methods.

6 Result Analysis

6.1 Pure LLM vs. Naive RAG

In this section, we analyze the impact of integrating retrieval capabilities into language models by comparing the performance of LLM-based generation (Naive) with the Naive RAG approach. For the Naive RAG baseline created in our experiments, the final answers are generated exclusively using either LLaMA-3-8B or GPT4o-mini as the underlying language model. This ensures a consistent evaluation framework and isolates the effect of retrieval augmentation on the overall system performance.

Table 2 summarizes the results:

Table 2: Comparison of Pure LLM and Naive RAG Performance

Model	False (%)	True (%)
LLaMA-3-8B Naive	88.18	11.82
LLaMA-3-8B Naive RAG	76.66	23.34
GPT4o-mini Naive	65.93	34.07
GPT4o-mini Naive RAG	64.65	35.35

6.1.1 Analysis

For both LLaMA-3-8B and GPT4o-mini, the integration of retrieval (Naive RAG) improves the accuracy of True responses compared to pure LLM-based answers. Specifically:

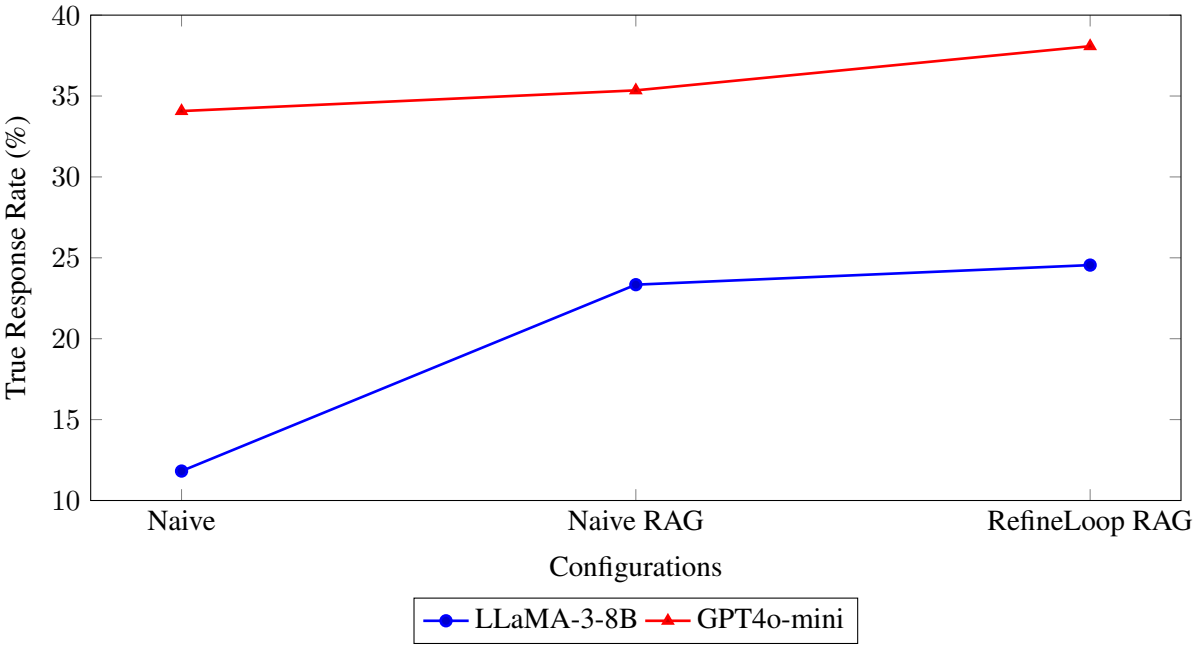

Figure 3: Trend lines of True response rates across configurations for both models.

- LLaMA-3-8B Naive RAG sees an improvement in True responses from 11.82% to 23.34%, highlighting the significant benefit of integrating external knowledge.

- GPT4o-mini Naive RAG also shows a marginal increase in True responses from 34.07% to 35.35%. The smaller gain suggests that GPT4o-mini's inherent capabilities handle some queries well without retrieval.

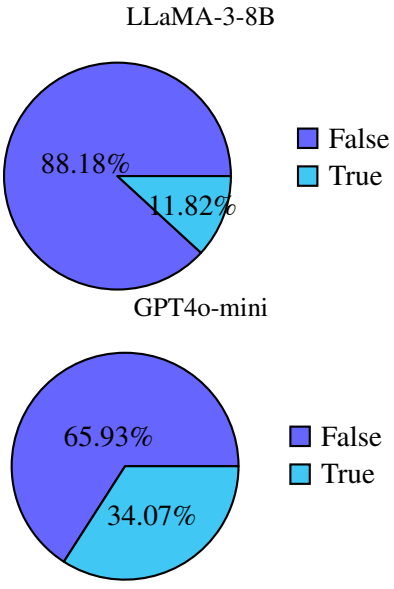

Figure 4: Comparison of False and True response rates for Pure LLM and Naive RAG.

6.2 Naive RAG vs. RefineLoop RAG

To assess the benefits of incorporating iterative feedback mechanisms, we conduct a comparative analysis between the Naive RAG baseline and our proposed RefineLoop RAG method. The Naive RAG baseline relies on a single-step retrieval process without any refinement, while RefineLoop RAG introduces a feedback-driven iterative query optimization loop. This comparison aims to quantify the improvements achieved through iterative query refinement in terms of model performance, particularly focusing on the percentage of True responses.

6.2.1 Analysis

- LLaMA-3-8B: RefineLoop RAG improves True responses from 23.34% to 24.55%, demonstrating modest gains due to feedback-based query refinement.

- GPT4o-mini: RefineLoop RAG achieves a more notable improvement, increasing True responses from 35.35% to 38.08%. This indicates that iterative refinement has a stronger impact on smaller models like GPT4o-mini.

To evaluate the effectiveness of the iterative feedback mechanism, we calculated the percentage improvement of RefineLoop RAG over Naive RAG for both LLaMA-3-8B and GPT4o-mini models. As shown in Figure 5, the results demonstrate consistent enhancements across all models:

Model	False (%)	True (%)
LLaMA-3-8B Naive RAG	76.66	23.34
LLaMA-3-8B RefineLoop RAG	75.45	24.55
GPT4o-mini Naive RAG	64.65	35.35
GPT4o-mini RefineLoop RAG	61.92	38.08

Table 3: Comparison of Naive RAG and RefineLoop RAG Performance

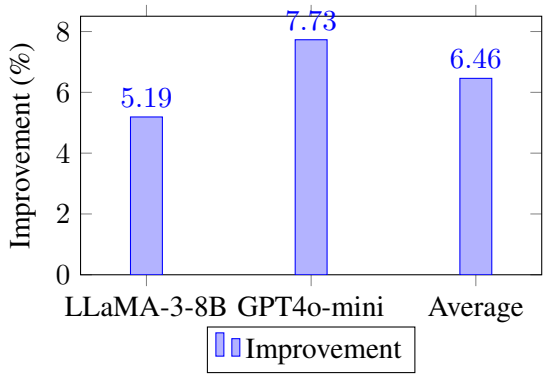

Figure 5: Percentage improvement of RefineLoop RAG over Naive RAG.

- **LLaMA-3-8B**: A 5.19% improvement is observed with RefineLoop RAG compared to Naive RAG. This demonstrates that even smaller models benefit from iterative query optimization, leveraging the refined process to enhance retrieval-augmented generation.
- **GPT4o-mini**: RefineLoop RAG achieves a 7.73% improvement over Naive RAG, indicating that larger models can significantly benefit from external knowledge integration and iterative refinement, enhancing their performance further.
- **Average Improvement**: The overall improvement across both models is 6.46%, highlighting the general effectiveness and adaptability of the proposed iterative feedback mechanism in boosting retrieval-augmented systems.

The larger improvement observed for GPT4o-mini suggests that iterative refinement compensates for its limited capacity by optimizing query formulations and retrieved content. In contrast, LLaMA-3-8B, as a larger model, benefits less from refinement, as its intrinsic generative capabilities already handle queries with moderate robustness. These findings highlight the potential of RefineLoop RAG in improving smaller and more resource-efficient models, making it particularly suitable for lightweight applications or scenarios with constrained computational resources.

6.3 Model Comparison

To explore the impact of model size and architecture on the performance of retrieval-augmented generation (RAG) systems, we compare the results of two distinct models, LLaMA-3-8B and GPT4o-mini, across three configurations: Naive (pure generative), Naive RAG (single-pass retrieval integration), and RefineLoop RAG (iterative query optimization with feedback). This analysis aims to highlight how different model architectures respond to retrieval augmentation and iterative refinement, providing insights into the scalability and adaptability of RAG frameworks. Table 4 summarizes the detailed results:

Table 4 summarizes the results:

Model	Configuration	False (%)	True (%)
LLaMA-3-8B	Naive	88.18	11.82
	Naive RAG	76.66	23.34
	RefineLoop RAG	75.45	24.55
GPT4o-mini	Naive	65.93	34.07
	Naive RAG	64.65	35.35
	RefineLoop RAG	61.92	38.08

Table 4: Model Performance Across Configurations

6.3.1 Analysis

- Model Size: LLaMA-3-8B demonstrates lower True response rates across all configurations, likely due to its larger size requiring more precise context to operate effectively.
- Iterative Gains: GPT4o-mini benefits more from iterative refinement, suggesting smaller models can leverage external feedback more effectively.
- RAG Effectiveness: Both models show clear gains with retrieval augmentation, validating the importance of external knowledge integration.

6.4 Question Classification

The performance of retrieval-augmented generation (RAG) systems can be impacted by the type of question. To explore the performance of RAG of answering different type of questions. We first separate the questions into 10 different topics, which

are 'Film', 'History', 'Other', 'Biology', 'Music', 'Literature', 'Sport', 'Geology', 'Law' and 'Nation'. The distribution of each topic is shown in the pie chart in Figure 6

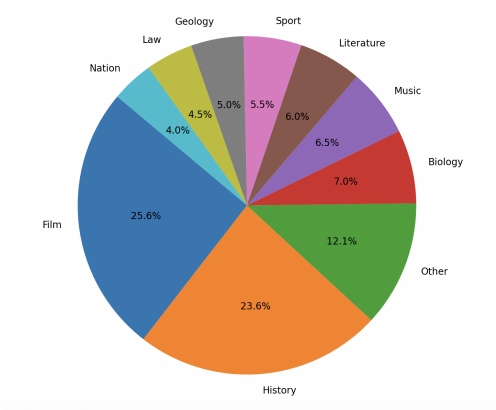

Figure 6: The distribution of each topic in the 500 questions

- Figure 7 and Figure 8 compare the performance of two models, GPT4o-mini and LLaMA-3-8B, in generating correct answers across various categories using three methods: naive, naive RAG, and improved RAG. Across both models, the improved RAG method consistently outperforms the other two, demonstrating the effectiveness of retrieval-augmented generation in enhancing model accuracy. Categories such as Film and Nation consistently show the highest performance, while Law, Literature, and Sport are notably challenging, with lower scores across all methods.

- When comparing the models, GPT4o-mini shows a significant edge in high-performing categories like Film and Nation, where the improved RAG method achieves the most correct answers. The difference between the naive and improved rag methods is more pronounced in GPT4o-mini, suggesting that this model benefits more from retrieval augmentation. Conversely, while LLaMA-3-8B exhibits lower peak performance in dominant categories, it offers more consistent results across all methods and categories, with smaller gaps between the naive RAG and improved RAG methods.

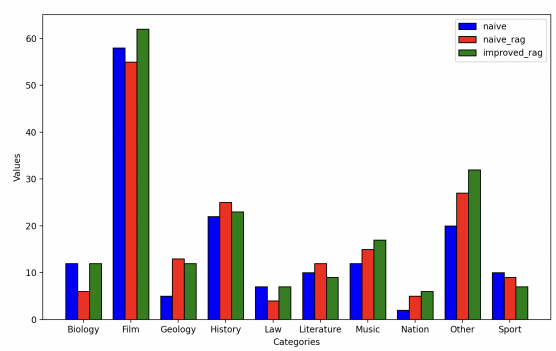

Figure 7: The number of correct answers separated by categories by using GPT4o-mini

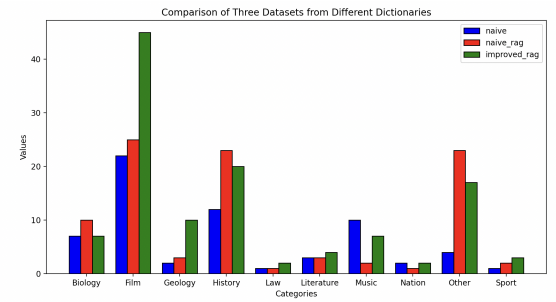

Figure 8: The number of correct answers separated by categories by using LLaMA-3-8B

7 Conclusion

This work presents a novel approach to enhancing RAG systems through iterative query refinement. By introducing dynamic evaluation loops, the method ensures continuous improvement of query formulations, addressing the limitations of static retrieval pipelines. The hybrid scoring mechanism, which combines generative and discriminative models, provides precise and actionable feedback on the relevance of retrieved content, facilitating optimal alignment between user queries and external knowledge.

Extensive experiments with models like LLaMA-3-8B and GPT4o-mini demonstrate the effectiveness of this iterative approach, achieving significant improvements in retrieval accuracy and answer generation for knowledge-intensive tasks. The integration of iterative refinement not only reduces reliance on manual query adjustments but also establishes a framework for robust, adaptable RAG systems capable of handling complex, multi-step reasoning processes.

This study underscores the transformative potential of combining retrieval, evaluation, and refinement loops in advancing the state-of-the-art in

natural language understanding and generation. Future directions include extending this framework to multi-modal applications and exploring reinforcement learning strategies for further optimization, paving the way for more intelligent, self-improving knowledge systems.

8 Limitation and Future Work

The proposed iterative query refinement framework improves retrieval accuracy but increases computational overhead, limiting scalability in real-time or resource-constrained settings. Its hybrid scoring mechanism depends on the quality of underlying models, and inaccuracies may propagate inefficiencies. The reliance on a static knowledge base restricts real-time applicability, and its generalizability to other domains or multi-modal tasks remains untested.

Future work should optimize the process for reduced latency, enhance the scoring mechanism with advanced techniques, and integrate dynamic, multi-modal knowledge bases. Broader validation across diverse datasets and tasks will further demonstrate its adaptability and utility, enabling more efficient and intelligent RAG systems.

9 Contributions

- The topic of the project was collectively brainstormed and finalized by all team members.
- Logan was responsible for writing and organizing the codebase, ensuring its functionality and scalability.
- Zimo and Yongqi were responsible for running experiments and analyzing the results.
- All team members contributed to drafting, editing, and finalizing the project report.

References

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 874–880.

Omar Khattab, Ahmed Khalifa, Kareem Darwish, and Donald Metzler. 2021. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Martin Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Yunhao Shi, Wei Zhang, Hao Liu, and Linlin Zhao. 2022. Self-rag: Adapting retrieval-augmented generation for personalized contexts. *arXiv preprint arXiv:2210.00546*.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. 2024. Take a step back: Evoking reasoning via abstraction in large language models.