|  |
| --- |
| **Retrieval Augmented Generation Models:**  **Incorporating new architectures components** |
|  |
| **Elkaim Ethan, Karsenty Eva** and **Lasry Raphael**  Technion – Israel Institute of Technology  ethan.elkaim@campus.technion.ac.il  [ethan.elkaim@technion.ac.il](mailto:ethan.elkaim@technion.ac.il) , [evakarsenty@campus.technion.ac.il](mailto:evakarsenty@campus.technion.ac.il) , [raph@campus.technion.ac.il](mailto:raph@campus.technion.ac.il) |
|  |
|  |

Abstract

Retrieval-Augmented Generation (RAG) enables Large Language Models (LLMs) to leverage external knowledge, enhancing their performance on knowledge-intensive tasks. This paper tries to introduces a novel approach of RAG capable of delivering good results while optimizing performance time. To achieve this, we implemented and evaluated three distinct models: a RAG-ANN model, a Hierarchical RAG model, and a Wikipedia-RAG model. Our experimental results show a slight performance improvement for the RAG-ANN and Hierarchical models, given that they utilize similar resources (dataset and LLM) and that the baseline RAG already delivers positive outcomes. These findings highlight the potential of novel architectures to enhance the indexing and retrieval stages of the standard RAG framework.

Introduction

Large Language Models (LLMs), e.g. Cohere(), have shown strong emergent abilities and achieved convincing performance in various NLP tasks (Wei et al., 2022a; Zhao et al., 2023). However, existing LLMs usually produce unreliable responses due to the hallucination and out-of-date memory (Ji et al., 2023; Bang et al., 2023; He et al., 2023), which challenges the reliability of LLMs. Retrieval-Augmented Generation (RAG) (Guu et al., 2020; Lewis et al., 2020; Gao et al., 2024) shows a promising approach to enhancing the response accuracy of LLMs by making them assess external knowledge (Asai et al., 2024). These RAG models usually feed the retrieved evidence as context and ask LLMs to answer the given question by in-context learning (Ram et al., 2023a). These models have demonstrated their capability in producing more convinced responses by leveraging retrieved knowledge to update the parametric knowledge of LLMs (Yu et al., 2023b). However, numerous errors persist, and some data remain outdated, revealing a significant flaw in these traditional RAG models.

Building on the core components of this technology, our goal is to dive deeply into two of the key elements of RAG (indexing and retrieval) to develop new approaches that enhance accuracy and improve search efficiency. So far, the combination of active learning and RAG has been explored (), but without fully investigating the variety of methods and combinations available. Additionally, we addressed the issue of data completeness in augmented prompts within our RAG, as well as the challenge of handling large dataset downloads, which can be time-intensive and require substantial storage. How can the data be as comprehensive as Wikipedia's database? While some researchers have developed RAG models specialized in specific domains (), achieving high precision with specialized datasets, their exploration often stops there.

Methodology

In this section, we will describe the three architectures models built for our study. The first two models focus on optimizing the indexing phase, while the third model places more emphasis on improving the retrieval phase by leveraging Wikipedia-style hierarchical structures to rank and retrieve contextually relevant passages more effectively. For each model, we used a consistent chatbot (cohere).

**Data collection and Preprocessing** For our RAG-ANN model, we choose to use the dataset: "ErikCikalleshi/new\_york\_times\_news\_2000\_2007" from the Hugging Face platform. In fact, it covers a wide range of topics from various fields, which allows our RAG model to remain general rather than specialized in a single domain.

This approach allows the model to address a broad array of subjects and maintain relevance across diverse topics. Similarly, for our hierarchical model, we based our approach on the “cnn\_dailymail” dataset, which also covers multiple topics.

However, we faced limitations with these first two datasets due to their size and restricted access to comprehensive data. To address this, we implemented a third model using Wikipedia’s extensive database and utilizing a Named Entity Recognition (NER) model to extract relevant keywords from the query and then directly accessing the corresponding Wikipedia pages. This approach ensures that, unlike the first two datasets, we do not need to download the entire Wikipedia dataset; instead, we dynamically retrieve only the necessary pages, providing comprehensive coverage while avoiding issues of excessive dimensionality.

**Preprocessing of the data**  For our RAG-ANN model, we extracted the `content` field from the dataset, containing the main text data for processing. We also add the possibility to apply a semantic chunking on our data. In this case, the text is then segmented into coherent sections by analyzing semantic similarity between units. Using the 90th percentile (our logical breakpoint threshold), chunks are split where similarity scores are among the lowest 10%, indicating significant topic shifts. This ensures each chunk maintains contextual relevance for better retrieval and processing.

For the Hierarchical model, we focused on segmenting and summarizing the data using a two-layer approach to capture both broad context and finer details. First, we split the data into larger sections that represent distinct articles or paragraphs, keeping the article field from the ‘cnn\_dailymail’ dataset. Each section is then summarized to create high-level representations that are used in the summary-level index. Within each section, we further divided the content into smaller, coherent chunks, allowing more detailed indexing and retrieval. This two-tiered structure enables the model to retrieve both general overviews and precise information, enhancing relevance during response generation.

Finally, for the Wikipedia-based model, we leveraged a Named Entity Recognition (NER) model to dynamically extract relevant keywords from the query. Based on these keywords, we accessed corresponding Wikipedia pages using the ‘wikipediaapi’ library, retrieving only the most pertinent information rather than downloading a large dataset. After accessing the pages, we performed preprocessing by breaking down each article into distinct paragraphs and filtering these based on their relevance to the query context. In cases where paragraphs were too lengthy, we applied summarization to condense the content while retaining core information. This targeted approach not only reduces dimensionality but also ensures the data remains highly relevant to the user's query, offering a dynamic, efficient alternative to static datasets.

**The RAG-ANN Model** As the classical RAG, our model is a part of a bigger pipeline:

It begins with a user submitting a query via a Cohere-powered chatbot. This query is embedded using the ‘paraphrase-MiniLM-L6v2’ embedding model from the SentenceTransformers library. This model provides compact, high-quality sentence embeddings optimized for semantic similarity tasks.

After loading and embedding the dataset, the embeddings were indexed using FAISS to build an Approximate Nearest Neighbors (ANN) search index. FAISS allows us to incorporate multiple steps to optimize our search using various methods. In our case, we utilized it to enable us to partition the index into Voronoi cells.

Using this approach, we process a query vector by first identifying the cell it belongs to. We then use the `IndexFlatL2` to search within that cell, comparing the query vector to all other vectors within the same partition.

Therefore, this strategy allows us to narrow the search scope.

To implement this, we first initialize our index using `IndexFlatL2`. However, in this setup, the L2 index serves as an intermediate step, which is subsequently used within the partitioning index, `IndexIVFFlat`. The retrieval system leverages FAISS's IndexIVFFlat to efficiently fetch the most relevant documents based on the query embeddings.

Here's some details of our implementation:

- The embeddings were grouped into 100 clusters (`nlist=100`) to accelerate the search process, while avoiding choosing a number that is too high relative to the size of the dataset used.

- When a query is provided, it is embedded and searched against the FAISS index to retrieve the top 3 most relevant documents based on cosine similarity (via L2 distance approximation).

The retrieved documents are combined to augment the query, creating a context-rich prompt. Finally, Cohere’s LLM generates a detailed response based on the augmented prompt, which is sent back to the user through the chatbot. No advanced re-rankers or additional layers were applied in this implementation because the innovation of this RAG lies in combining the advantages of RAG with those of ANN to streamline the retrieval process.

**The Hierarchical model** This model incorporates a two-tiered indexing and retrieval system, designed to capture and differentiate between higher-level summaries and more granular details within documents. It also begins with a user query submitted via a Cohere-powered chatbot. The query is first embedded using ‘paraphrase-MiniLM-L6v2’ to generate compact, semantically rich embeddings.

For the indexing phase, we implement a two-level Pinecone index structure:

1) Summary-Level Index - Each document is segmented into distinct sections (e.g., articles or long paragraphs) which are then summarized. These summaries form the basis of the first-level index, allowing the model to quickly identify which sections of the dataset are likely to contain relevant information. This initial summary-level indexing provides a broad but efficient filter, narrowing down the search to only those segments relevant to the query’s context.

2) Chunk-Level Index - Within each summarized section, we split the content into smaller, coherent chunks. Each chunk is independently embedded and stored in a second-level index. When a relevant summary is identified in the first step, the model retrieves the most pertinent chunks within that section, providing the user with both the overarching context and specific details.

*Retrieval and Response Generation:*  
During retrieval, the model first queries the summary-level index to obtain top summaries relevant to the query. Then, it performs a second query within the chunk-level index to retrieve the most pertinent details within these summaries. This hierarchical retrieval setup enables us to efficiently handle larger datasets by focusing only on specific document sections and their granular details.

To enhance relevance, we apply a **re-ranking step** that scores the retrieved chunks based on their similarity to the query using the ‘CrossEncoder’ model. After re-ranking, the top-ranked chunks are combined to create a context-rich prompt, which is then used to generate a response with Cohere’s LLM. This re-ranking ensures the final prompt is both contextually accurate and focused on the user’s intent, improving the specificity and relevance of the responses.

**The Wikipedia-based model** This model innovates by dynamically sourcing information from Wikipedia pages instead of relying on pre-loaded, static datasets. The model’s pipeline begins by extracting keywords from the user’s query using a Named Entity Recognition (NER) model. These keywords guide the retrieval of relevant Wikipedia pages via the ‘wikipediaapi’ library, allowing the model to directly access the most current and comprehensive information on any given topic without downloading the entire Wikipedia dataset.

*Keyword Extraction and Page Retrieval:* The NER model identifies key terms from the user’s query, which are then used to search for corresponding Wikipedia pages. This targeted retrieval method mitigates the issue of dataset size and dimensionality by only retrieving pages related to the identified entities or topics.

*Hierarchical Indexing and Summarization:* Once a page is retrieved, it is broken down into paragraphs or sections. Each paragraph is embedded and stored in a Pinecone chunk-level index. To facilitate efficient access, the Wikipedia model also creates a summary-level index, where each section of the retrieved page is summarized, providing an overarching view of each article's content.

***Response Generation:***  
For each query, the model first retrieves relevant summaries from the summary-level index and then performs a more detailed search in the chunk-level index. The retrieved chunks are re-ranked using a ‘CrossEncoder’ to ensure relevance. This re-ranked set of paragraphs forms a comprehensive, context-rich prompt for Cohere’s LLM, enabling detailed responses based on the most pertinent Wikipedia content available at the time.

Experiments

In this section, we will describe the experimental setups conducted and the results obtained for each of our three models.

**Evaluation metrics** One of the challenges that arises when building a new RAG model architecture is the evaluation of the model. Indeed, without a document containing our "ground truth," it becomes impossible to measure the effectiveness of our models using standard metrics such as accuracy. To address this issue, we propose different types of evaluation metrics. The first allows us to evaluate the semantic coherence between a query and a response using cosine similarity. It generates embeddings for both the query and the response through a pre-trained language model and then calculates the cosine similarity between them. The score ranges from -1 (completely opposite) to 1 (highly coherent). We chose to create this metric because it focuses on the semantic relevance of the generated response, ensuring that it aligns with the query's meaning. It helps assess how well the model generates contextually accurate and meaningful answers.

Our second metric will focus more on the accuracy of our responses. This metric compares the similarity between the original response from our RAG model and several reformulated responses. The process involves generating alternative formulations of the initial query using a language model like Cohere, then retrieving responses for each reformulated query. Next, embeddings of these responses are generated using a pre-trained SentenceTransformer model, and cosine similarity is computed between the original and reformulated responses. The advantage of this metric is that it ensures the RAG model generates consistent and contextually relevant answers, even when the query is phrased in different ways. This approach helps assess the model’s robustness and flexibility in addressing various query phrasings.

**Baselines** The baseline consists of Vanilla RAG model. We use Cohere as backbone LLMs to implement the baseline model.

We utilize a simple prompt to our LLM, just asking him to respond to our query based on the information he had. Then we follow previous work (Wei et al., 2022b) and implement a model that generates the reasoning process to answer the question. Concerning the basic RAG model used as baseline, we choose the RAG model learnt during our tutorial.

**Implementation Details** To evaluate our models in a consistent and coherent manner, we used a varied selection of queries that remained the same across multiple models. This ensured that the model's results were not biased by specific queries.

**The RAG-ANN model** Under the name RAG-ANN model, we tested various parameters in our model. First, we observed that the results of our standard RAG-ANN model, without using semantic chunks, were highly satisfactory, achieving an average similarity score of approximately 0.80 using our query reformulation evaluation metric, and an average of around 0.75 using our semantic coherence evaluation metric. When we incorporated semantic chunking into our data preprocessing, we achieved an average score of approximately 0.70 using our query reformulation metric, with significant variations depending on the type of query. Additionally, we obtained a score of around 0.80 using our semantic coherence metric. This suggests that the importance of semantic chunking within our model varies based on the query type (details of our experiments and results scores are presented in the appendix A). Finally, we observed that while semantic chunking increases data preprocessing time, it reduces retrieval time. In contrast, the standard RAG-ANN model requires less time for data import and processing but takes slightly longer during the retrieval phase.

It is also important to note that our results show a certain improvement compared to our Vanilla RAG model. For example, when looking at query 1, we observe a semantic coherence score of 0.75 using a standard RAG model, whereas with our RAG-ANN model, without applying semantic chunking, we already achieve a semantic coherence score of 0.79. This demonstrates an improvement in our evaluation metrics compared to the standard RAG model.

**The Hierarchical Model** TODO

**The Wikipedia-based model** TODO

Conclusion And Discussion

This paper introduces several models, each featuring a modified and improved component of the standard RAG. Our experiments show that RAG-ANN, Wikipedia-RAG, and Hierarchical-RAG highlights the potential of enhancing traditional RAG architectures through innovative indexing methods, enriched datasets, and hierarchical retrieval strategies, ultimately improving both the accuracy and efficiency of information retrieval in various contexts.

However, certain limitations remain, such as reduced performance on overly broad or ambiguous queries and increased preprocessing times, which may impact scalability in real-time applications and potentially be a direction for further improvement of the RAG model.

**5 Appendix**

References

GitHub’s link for the project’s code:

<https://github.com/ethanelkaim/RAG>

XU, Zhipeng, LIU, Zhenghao, LIU, Yibin, *et al.*

Activerag: Revealing the treasures of knowledge via active learning. *arXiv preprint arXiv:2402.13547*, 2024.

MAHBOUB, Ali, ZA'TER, Muhy Eddin, AL-RFOOH, Bashar, *et al.*

Evaluation of semantic search and its role in retrieved-augmented-generation (rag) for arabic language. *arXiv preprint arXiv:2403.18350*, 2024.

ROYCHOWDHURY, Sujoy, SOMAN, Sumit, RANJANI, H. G., *et al.*

Evaluation of RAG Metrics for Question Answering in the Telecom Domain. *arXiv preprint arXiv:2407.12873*, 2024.

1. Experimental Detailed Results for the RAG-ANN model

In this section we present a table that show the performance of our RAG-ANN model, using our two different evaluation metrics and multiple type of queries. TODO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Query / Type of RAG model and evaluation metric | RAG-ANN without semantic chunking  Semantic coherence | RAG-ANN without semantic chunking  Reformulation query | RAG-ANN with semantic chunking  Semantic coherence | RAG-ANN with semantic chunking  Reformulation query |
| How many jobs were added on Long Island in 2001? |  |  |  |  |
| In 2003 Turkey were waiting for the Bush administration to answer their demand for an economic aid package to ensure their participation in a war with Iraq. What was the amount of the economic aid package? |  |  |  |  |
| How much was the prize for the winners of the first Westchester Prize for New Works? Among how much Westchester arts organizations was it shared equally? |  |  |  |  |
| Who is Edith Friedman ? |  |  |  |  |