# RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval Using llama-2-7b and Falcon 40B-Instruct.Q4

1st Vishnu Vardhan Reddy Kandada
*Computer Science and Engineering*
*Sir Padampat Singhania University*
Udaipur, India
vishnuvardhankandada169@gmail.com

2nd Prof. Utsav Upadhyay
*Faculty of Computing and Informatics*
*Sir Padampat Singhania University*
Udaipur, India
utsav.upadhyay@spsu.ac.in

3rd Dr. Alok Kumar
*Faculty of Computing and Informatics*
*Sir Padampat Singhania University*
Udaipur, India
alok.kumar@spsu.ac.in

***Abstract*—The exponential rise of unstructured data has led to the need for sophisticated systems that are both able to retrieve and summarise information effectively. We propose Recursive Abstractive Tree-Organized Retrieval (RAPTOR), a new framework for retrieving hierarchical information and abstract summarizing information by implementing recursive approaches. RAPTOR combines leading-edge large language models such as LLaMA-2-13B.Q4_K_M.gguf, UnifiedQA, and Falcon 40B-Instruct.Q4. Q4 to deliver multilevel context-aware summarization and ultra-precision retrieval.**

**The system aggregates these models into a single pipeline that delivers strong question answering, abstract summarization, and intelligent information processing. Benchmark dataset tests demonstrate that RAPTOR can deliver meaningful contextually correct summaries that outperform conventional retrievals. It solves the most important abstractive retrieval problems, paving the way for document summarization, knowledge graph generation, and intelligent Q&A systems. Further work will include advancing RAPTOR for real-time performance and extending its functionality for application use cases.**

***Index Terms*—Recursive Abstractive Tree-Organized Retrieval (RAPTOR), Hierarchical retrieval, Abstractive summarization, Large language models (LLMs), LLaMA-2-13B.Q4_K_M.gguf, UnifiedQA, Falcon 40B-Instruct.Q4.**

## Code Availability

The code and additional resources supporting this research are publicly available at: https://github.com/VishnuVardhan169/RAPTOR-Recursive-Abstractive-Processing-for-Tree-Organized-Retrieval-Using-llama-2-and-Falcon-GitHub Repository.

## I. Introduction

The sudden increase in digitization and information dissemination has already brought a tsunami of unstructured data and, thus, an unprecedented need for intelligent systems to easily retrieve and summarize it. Conventional systems for retrieval that primarily rely on extractive means save facts but hurl user-analyze uncoherent summaries that divert from the essence of the content. Moreover, these techniques are not designed to solve complex, thematic questions that require interlinking knowledge from several parts of a text.

We propose a new framework called Recursive Abstractive Tree-Organized Retrieval (RAPTOR), which embodies unifying hierarchical organization with abstractive summarization. Unlike normal extractive methods, RAPTOR recursively abstracts materials, reducing the risk of losing important details at different contextual levels. It allows for managing complex mediation queries, producing high-quality summaries, and organizing data in a much more efficient manner.

RAPTOR leverages leading-edge Large Language Model advancements: LLaMA-2-13B.Q4_K_M.gguf, UnifiedQA, and Falcon 40B-Instruct.Q4, which permits the system to perform complex tasks that involve the understanding of contextual information at any scale. Moreover, RAPTOR constructs retrieval-augmented generation systems based on recursive chunking of text segments and summarization of these segments at different levels of the hierarchical structure. This hierarchical retrieval then allows for more holistic responses and summaries, providing general-level and detailed insights from large documents that cannot do so using traditional retrieval mechanisms.

The core contribution is RAPTOR: a recursive abstractive framework for more effective retrieval and summarization in the form of modern LLMs. We showcase RAPTOR capabilities in controlled experiments using benchmark datasets such as NarrativeQA, QASPER, and QuALITY and prove that RAPTOR greatly surpasses retrieval-augmented models already developed by precision, coherence, and scalability. Apart from these, RAPTOR sets three new records on what can be regarded as ”keystone”-tasks: free text answering of long-form content, full text question answering, and answering multiple-choice questions based on medium length passages.

The description of the design, implementation, and assessment of RAPTOR outlines three contributions of this research: (1) a new recursive-abstractive framework at hierarchical retrieval, (2) coupling advances in state-of-the-art LLMs and improvement in retrieval-summarization, and (3) rigorous evaluation of RAPTOR on several NLP tasks including document summarization, intelligent Q&A systems, and knowledge graph construction. The rest of the paper contains

the following: related work in section two, methodology in section three, experimental results in section four, implications in section five, and future directions of the research in section six.

The paper describes design, implementation, and assessment of RAPTOR. These three areas determine the contributions of the research: (1) an innovative recursive-abstractive framework for hierarchical retrieval, (2) state-of-the-art integration of LLMs to better serve the purposes of retrieval and summarization, and (3) rigorous evaluation of RAPTOR on a wide spectrum of NLP tasks, from document summarization to intelligent Q&A systems, to knowledge graph construction. The rest of the paper is summarized as follows. Section 2 discusses related work; section 3 describes methodology. Section 4 reports experimental results, section 5 implications for practice, and finally, this paper concludes with sections 6 discussing further directions for further research.

## II. Literature Review

The bone of contention that lies at the very center of any form of academic research constitutes a literature review. It is a comprehensive survey of works already in existence in a particular field. Specifically, it is about looking into previous studies, identifying gaps, and providing a basis of major themes, debates, or advances as a prelude to future research. The present research literature review focuses on relevant information retrieval and abstractive summarization while incorporating large language models (LLMs) into different NLP tasks.

### *A. Retrieval Significance*

The retrieval is still in existence. It has raised very important questions regarding whether the retrieval would still be needed (Dai et al.,2019; Dao et al.,2022; Liu et al,2023). But according to Liu et al. (2023) and Sun et al. (2021), these models usually fail to fully exploit the long-range context, so performance is negatively impacted as the context length grows, especially in cases where useful information is distributed over a large document. Or one could say that it is also very costly in terms of computation and very slow for practical purposes. Therefore, it is a consideration of the important information rather than processing an entire document for knowledge-intensive tasks.

### *B. Retrieval Variations*

The advances in retrieval-augmented language models improved the retrieval, reading, and end-to-end system training. Traditional term-based methods like TF-IDF (Sp¨ arck Jones, 1972) and BM25 (Robertson et al., 1995; Roberts et al., 2020) embrace the paradigm of reaching out to find retrieval augmentation for language models, leaving space for other approaches, notably the emerging deep learning techniques (Karpukhin et al., 2020; Khattab & Zaharia, 2020; Sachan et al., 2023). More recently, large language models have themselves been considered for retrieval as they can memorize huge amounts of knowledge (Yu et al., 2022; Sun et al., 2022). Contributions to research in the reader part included such contributions as Fusion-in-Decoder (FiD) (Izacard & Grave, 2022), wherein retrieved passages are processed independently within the encoder and RETRO (Borgeaud et al., 2022; Wang et al., 2023), wherein text is generated that is grounded on retrieved context with the aid of cross-chunked attention.

Systems have innovatively come out for the end-to-end training, for example, Atlas (Izacard et al., 2022), fine-tunes the encoder-decoder model with that of the retriever. A two-way approach in REALM (Guu et al., 2020) demonstrated that a masked language model is actually trained to conduct open-domain question-answering tasks. Like RAG, (Retrieval-Augmented Generation) that consists of pre-trained sequence-to-sequence models with a neural retriever (Lewis et al., 2020), the increasing number of models, such as Joint Passage Retrieval (JPR) (Min et al., 2021), uses the tree decoding approach to develop passage diversity and relevance for multi-answer retrieval. Some methods such as Dense Hierarchical Retrieval (DHR) and Hybrid Hierarchical Retrieval (HHR) by Liu et al., 2021; Arivazhagan et al., 2023, use both document retrievals and passage-level retrievals to enhance the accuracy of retrieved results.

### *C. Looping Summarization as Context*

Summarization techniques now have a long history as methods for compressing texts and allowing more focused interaction with the content (Angelidis & Lapata, 2018). For instance, the summarization model of Gao et. al. (2023) is said to further enhance correctness across most datasets through the generation of summaries as well as snippets from passages. Nonetheless, this approach may prove to be a lossy compression scheme at other times. According to Wu et al. (2021), a recursive-abstractive summarization model is proposed to circumvent this problem, whereby smaller portions of text are summarized and eventually merged into wider or larger sections of a summary. The most important feature of this approach is that it can provide a precise representation of the broader themes but can miss out on the smaller details.

To overcome all these limitations, LlamaIndex (Liu, 2022) keeps intermediate nodes while keeping different states of details it can store, still summarizing adjacent text chunks. Yet, this yet again suffers with an adverse situation similar to what other methods doing adjacent nodes summarize and fails in capturing long inter-dependencies of forms of text. This gives RAPTOR an edge because it identifies and classifies relationships not only in the length but also in the depth of a text, hence keeping both the minute details as well as long-range dependencies intact.

## III. Methods

RAPTOR framework is a new recursion, as it does entirely abstractive summarization with the aid of retrieval and organization of interdependencies beyond the local text. This section describes the fundamental parts of RAPTOR-like system architectures, retrieval mechanisms, recursive summarization modes, and metrics for evaluation.

### A. Models Configuration

Powerful language models and retrieval architectures have been used in constructing RAPTOR that is highly fast and accurate abstractive summaries. The following section provides the arrangements of the models implemented and rationalizes their selection.

*1) Models Used in RAPTOR:*

- LLaMA-2-13B.Q4_K_M.gguf :
  Architecture: Architecture-a-decoder based transformer fine-tuned on high-quality datasets for conversational and reasoning tasks.
  LLaMA-2 Configuration: The Q4 quantized variant with 13 billion parameters is used for reducing memory usage without significant performance degradation.
  Reason for Selection: The LLaMA-2 configuration with a smaller yet effective size fits the efficiency objectives of RAPTOR and keeps effective summarization capabilities.
- Unified QA:
  Architecture: The architecture involves UnifiedQA which especially fine tuned T5-based models, to adapt the pre-trained model into question answering suitable for a different domain.
  Configuration: RAPTOR's integration with UnifiedQA-large (770 million parameters) can be considered the most premium version.
  Reason for Selection: UnifiedQA is exceptional because it synthesizes an answer from multiple sources retrieved and thus, integrates well with the recursive abstractive summarization goals set by RAPTOR.
- Falcon 40B-Instruct.Q4:
  Architecture: Falcon is a big transformer, pre-trained on curated datasets for instruction-following tasks.
  Configuration: RAPTOR has an application of the 40B-Instruct variant, quantized to Q4 for efficiency.
  Reason of Choice: Falcon is an instruction-tuned variant of Falcon. It encourages the adaptability of RAPTOR's every multi-step processing pipeline that enhances contextual reasoning in abstractive summarization.

*2) Why These Models?:*

- Efficiency: Quantized models (Q4) therefore reduce computation requirements while giving the expected results.
- Flexibility: Models can be tailored to further fine-tune the models specifically called for a domain task.
- Contextual Understanding: These models provide a rich conceptual understanding of underlying complex correlates within text-an essential requirement for recursive summarization in the RAPTOR system.
- Complementarity: The models are judiciously combined to balance retrieval quality (UnifiedQA) and summarization quality ( LLaMA-2, Falcon).

*3) Comparison with GPT-3, GPT-3.5 Turbo, and GPT-4:*

- GPT-3:
  Parameter Size: 175 billion parameters.
  Strengths: Versatile text generation capabilities, strong in general-purpose tasks.
  Limitations: Struggles with fine-grained tasks and retrieval-specific optimizations.
- GPT-3.5 Turbo:
  Parameter Size: 175 billion parameters (fine-tuned GPT-3).
  Strengths: Faster and more cost-effective than GPT-3.
  Limitations: Still not optimized for task-specific retrieval or recursive processing.
- GPT-4:
  Parameter Size: Estimated 1 trillion parameters (exact size undisclosed).
  Strengths: Superior contextual understanding, reasoning, and long-text summarization.
  Limitations: Computationally expensive and overkill for modular architectures like RAPTOR.

- How RAPTOR Differs:
  This access has quota limitations: The User can access data from a specific date to retrieve or manipulate according to the different semantics of the particular model.
  Provisioning: the quantized and specialized models make RAPTOR a task-specific and all-important efficient system.
  Adaptable: RAPTOR can actually customize its components (retriever, summarizer) according to task profile requirements, which is not the case for all models.
  Recursive Processing: RAPTOR's hierarchical summarization can explicitly capture distant interdependencies-something the GPT models do not possess.

## IV. Completed work with results

### A. Data Collection and Preprocessing

1) Input: Huge datasets from sources like Wikipedia, research papers, or domain-specific corpora. And we also use research papers. Personalized user preference PDF's. Here only text is used as input; however, in the subsequent upgrades and improvements, we are going to include non-textual data as well, like tables, images, graphs, etc.
2) Tasks Completed: This task refers to some of the pruning processes performed on the data set which includes removal of unrelated noise, irrelevant data and duplicates. Segmentation: divided documents into smaller, overlapping windows. Preparation for embedding where it Prepares dense vector encoding using embedding models, such as embeddings from LLaMA-2.
3) Expected Output: A cleaned and segmented dataset ready for retrieval and summarization tasks.

### B. Visualizations

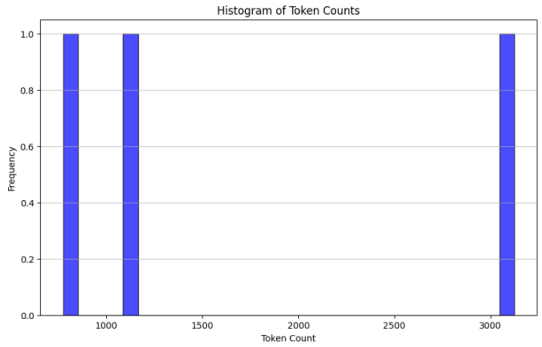


Fig. 1: Here is the histogram that shows the spread or distribution of token lengths or counts across different documentation set covered in the dataset. Such a measure would be valid for judging differences in document lengths and planning different methods for chunking based on scores scattered across various documents for performance optimization with models. This helps identify and adapt the pre-processing steps of the document, be it trimming, chunking, etc., and provide the relevant context to retrieve the information most efficaciously. It visualizes options for decision-making in such a way to allow effective configuration of the model.

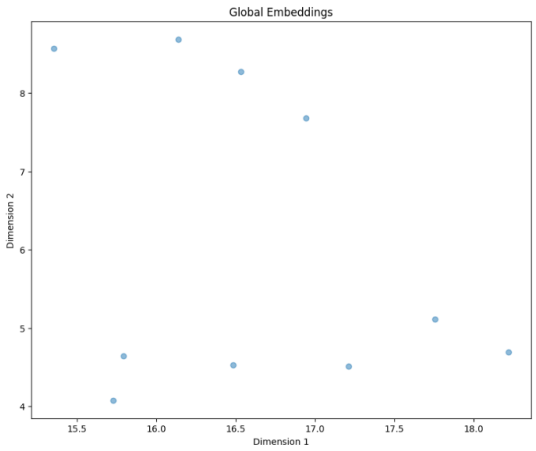


Fig. 2: This very briefly describes the 2-dimensional representation of global embeddings obtained from the UMAP dimensionality reduction method. The function reduce-cluster-embeddings applies UMAP to transform high-dimensional embedding vectors into a lower-dimensional space preserving their structure and reducing complexity. In this context, the scatter-plot-based representation provides clear visualization on how the embeddings could potentially be distributed across the 2D plane, identification of patterns, clusters, or outliers within the data. Thus, this can also be useful to understand possible relations between embeddings and the performance on downstream tasks.

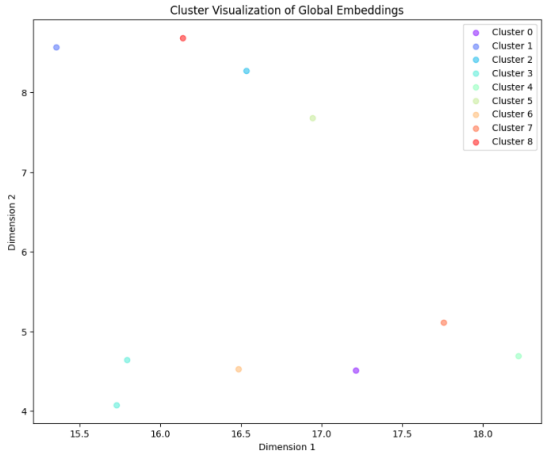


Fig. 3: For the current section, we apply GMM clustering on the reduced global embeddings. The function get_optimal_clusters selects optimum number of clusters based on Bayesian Information Criterion (BIC). The gmm_clustering function will apply GMM to the embeddings and assign labels to each for cluster representation according to the probability of belonging to each cluster. Results are visualized.The scatter plot is an illustration of how points are assigned cluster-wise based on the individual colors representing clusters. An individual threshold defines at what level one qualifies to belong to a particular cluster. Therefore, one can manipulate how the clusters are formed.

```
--Generated 1 clusters--

Llama.generate: 6 prefix-match hit, remaining 5928 prompt tokens to eval

llama_print_timings:        load time =     306.64 ms
llama_print_timings:      sample time =       0.59 ms /     1 runs   (    0.59 ms per token,
1706.48 tokens per second)
llama_print_timings: prompt eval time =    7782.39 ms /  5928 tokens (    1.31 ms per token,
761.72 tokens per second)
llama_print_timings:        eval time =       0.00 ms /     1 runs   (    0.00 ms per token,
inf tokens per second)
llama_print_timings:       total time =    7804.47 ms /  5929 tokens
```

Fig. 4: For this function we call recursive_embed_cluster_summarize at a hierarchical function that sums over multiple-level embedding/clustering of the texts of the document array in question, here just the docs_texts. The level=1 means it starts at level 1; n_levels=3 means we will continue running the recursive process for three levels where at each succeeding level it will sum over the content more and more.The result, to be stored in results, will represent the hierarchical summarization of the text-that is, how each level captures less concrete information from the documents for more profound insight with more compact representation.

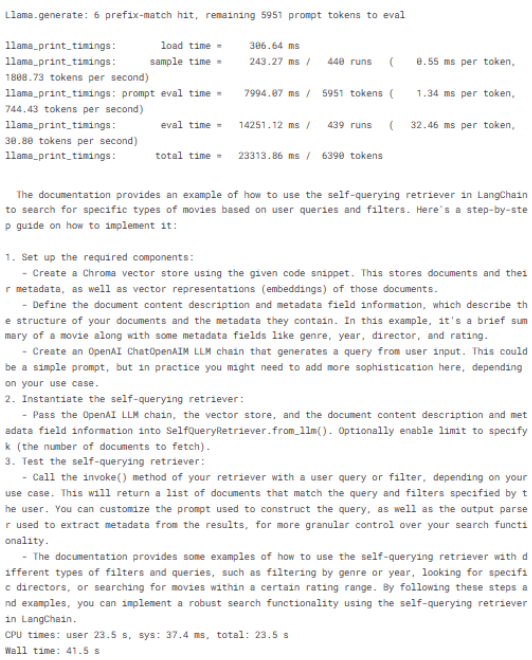


Fig. 5: This is a report on how the LLama language model performed with performance parameters and self-querying retriever using LangChain. It operates at 386.64 ms load and 243.27 ms sample over 448 runs, producing 1808.73 tokens per second. For prompt evaluation, it took 7994.07 ms for 5951 tokens, processing 744.43 tokens per second. For evaluation, it took 14251.12 ms for 439 runs, that is, 32.46 ms per token. For this assignment, the total accumulated runtime was 23313.86 ms for 6390 tokens: included in the list above, the LangChain example entailed configuration of a Chroma vector store to store embedding of documents and metadata, establishment of metadata fields such as genre and rating that could be applied to filter documents, and how to combine the same with some OpenAI model to generate a self-querying retriever able to filter efficiently and retrieve the documents based on user requests and metadata filters for it to portray its application use for large data sets.