# KV-RAPTOR: Scalable Tree-Structured Retrieval with KV Cache Compression for Question-Answering Systems

**João Gabriel J. da Silva**[1], **Sávio S. T. de Oliveira**[1], **Lucas Alexandria Alves** [1], **Nicolás Eiris**[2], **Arlindo R. Galvão Filho**[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Goiânia – GO – Brazil

[2]Panoplai
New York, N.Y.

{jgabriel,lucas.alves2}@discente.ufg.br

{savioteles, arlindogalvao}@ufg.br,

{nicolas}@panoplai.com

***Abstract.*** *This paper introduces KV-RAPTOR, a latency-optimized variant of the RAPTOR (Recursive Abstractive Processing for Tree-Organized Retrieval) pipeline for Retrieval-Augmented Generation (RAG). By integrating CacheGen, a compressed key-value (KV) cache reuse mechanism, into RAPTOR's tree-based index, we demonstrate that it is possible to reduce generation latency without sacrificing answer quality. We evaluate our method on both English and Portuguese datasets, showing consistent reductions in time-to-first-token and end-to-end latency while preserving performance across diverse linguistic and retrieval contexts.*

## 1. Introduction

Retrieval-Augmented Generation (RAG) has recently gained significant traction in natural language processing, with numerous studies and real-world applications leveraging its ability to enhance generative models through external information retrieval [Yu et al. 2024]. By exploiting trustworthy knowledge bases, RAG enriches the context provided to language models, thereby reducing non-factual or hallucinated responses [Taschetto and Fileto 2024].

As retrieval architectures evolve alongside the growth of large-scale knowledge bases, new methods have emerged to improve the quality and structure of retrieved contexts. Approaches such as HippoRAG [Jimenez Gutierrez et al. 2024], RAPTOR [Sarthi et al. 2024], and GraphRAG [Edge et al. 2025] exemplify advances in hierarchical organization, semantic clustering, and graph-based context structuring. These innovations aim to increase retrieval precision while reducing the cognitive and computational load on language models, resulting in more accurate and contextually grounded responses.

Despite these advances, LLMs' substantial computational and memory demands often require high-performance GPU servers. Even then, these resources can be strained by the sheer size of the models and the long text sequences they must process [Li et al. 2024]. As noted in [Sarthi et al. 2024], models tend to underutilize long-range

context and experience diminishing returns as input length grows, especially when relevant information is embedded in extended passages. Consequently, long contexts become expensive and slow to process, limiting their practical viability.

Key-Value (KV) cache management has emerged as a critical optimization technique for accelerating inference in response to the challenges of scaling LLMs to real-world, long-context, and real-time applications [Li et al. 2025]. Recent works such as CacheGen [Liu et al. 2024b], CacheBlend [Yao et al. 2025], and Minicache [Liu et al. 2024a] demonstrate how refined caching strategies can further enhance efficiency by optimizing memory usage, reusability, and compression of KV states, thus paving the way for scalable and performant LLM deployment.

In this context, we propose KV-RAPTOR, a method that extends the RAPTOR architecture by incorporating key-value (KV) caching mechanisms originally introduced in CacheGen. RAPTOR was selected as the foundation for our method due to its consistent performance among recent RAG approaches reported in the literature. While RAPTOR [Sarthi et al. 2024] offers a recursive indexing and retrieval framework that improves context relevance and scalability, it does not support cache reuse during generation. Our approach integrates compressed KV cache propagation within the decoding stages, allowing token representations to persist across retrieved segments. We further introduce parallel and batched generation routines, aiming to reduce end-to-end latency and time to first token across Portuguese and English datasets.

The main objective of this work is to design and evaluate an inference-efficient information retrieval pipeline that preserves answer quality while reducing processing time. Our main contributions are: (1) a modification of the RAPTOR decoding pipeline to support compressed KV cache reuse; (2) the design of parallel and batched indexing and inference strategies adapted to recursive retrieval structures; and (3) an experimental evaluation of the proposed method in multilingual scenarios, analyzing performance across varying retrieval depths and hardware constraints.

The remainder of this paper is organized as follows. Section 2 summarizes related work on retrieval architectures and caching techniques for large language models (LLMs). Section 3 describes the KV-RAPTOR architecture and its implementation. Section 4 outlines the datasets, indexing strategies, and evaluation methodology. Section 5 presents the experimental results. Finally, Section 6 shows the conclusions and directions for future work.

## 2. Background and Related Work

This section provides an overview of related work in areas connected to this paper's core contributions, namely retrieval-augmented generation (RAG) architectures, key-value (KV) cache reuse for inference acceleration, and methods that address the trade-off between retrieval quality and latency in LLM pipelines.

### 2.1. Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) was introduced by Lewis et al. [Lewis et al. 2020] to enhance generative language models with non-parametric memory by integrating external information retrieval. RAG pipeline comprises four main stages: knowledge base indexing, query processing and prompt construction,

information retrieval, and response generation. The indexing stage is performed offline and involves preprocessing the knowledge base by segmenting documents, extracting entities or summaries, and formatting the data for storage in a database. This indexed content is later accessed by the retrieval component during inference. During inference, the pipeline begins by processing the user query, which may involve classification, translation, paraphrasing, or embedding computation. The processed query is then used to construct a prompt aligned with the language model's constraints and retrieve relevant documents. The retrieval step uses information retrieval techniques to select the top-k segments from the indexed database. In the final generation stage, the retrieved content is passed to the language model, which synthesizes a response based on the query and retrieved context [Oliveira 2024]. This hybrid structure allows models to remain up-to-date without retraining, making it particularly effective for knowledge-intensive tasks.

Several studies have proposed architectural or retrieval-level enhancements to improve RAG pipelines. HippoRAG [Jimenez Gutierrez et al. 2024] introduced a biologically inspired memory hierarchy, combining hippocampal-style indexing with knowledge graph consolidation to enable dynamic retrieval over long contexts. GraphRAG [Edge et al. 2025], in turn, organizes retrieved chunks as connected entities within a local-global graph, improving semantic coherence across retrieved content. These works aim to refine retrieval precision and contextual relevance, two critical components in effective RAG design.

In the Brazilian context, [Taschetto and Fileto 2024] applied RAG to the ENEM national university exam, demonstrating measurable improvements across multiple disciplines and both text and multimodal data. Their work reinforces the role of RAG in multilingual and domain-specific evaluation benchmarks. Similarly, [Aquino et al. 2024] proposed a configurable RAG workflow to extract structured information from legal documents, underscoring the relevance of parameter tuning—such as chunking strategy, embedding model, and Top-K value—in practical RAG deployments.

### 2.2. Key-Value (KV) Caching

Key-Value (KV) caching is an important optimization technique for accelerating LLM inference. It involves storing intermediate transformer activations—specifically, the keys and values generated during self-attention—so that previously computed information can be reused across decoding steps. This reuse enables the model to avoid redundant attention computations over past tokens, performing only incremental updates for each new token. As a result, KV caching reduces both latency and memory usage during autoregressive generation [Jiang et al. 2024].

Recent studies have extended this approach by introducing compression and reuse strategies that further improve efficiency, particularly in long-context and retrieval-augmented generation scenarios. CacheGen [Liu et al. 2024b], CacheBlend [Yao et al. 2025], and Minicache [Liu et al. 2024a] each propose methods for compressing or structuring KV cache data to enable reuse across varying inputs and retrieval iterations. These methods contribute to lower time-to-first-token (TFFT) and support scalable deployment of LLMs in latency-sensitive and resource-constrained settings.

Our work builds upon RAPTOR [Sarthi et al. 2024], which constructs recursive, tree-based cluster indices using summarization for scalable document retrieval, and CacheGen [Liu et al. 2024b], which compresses and reuses cached activations to reduce decoding overhead. While these methods were developed independently, we integrate their core ideas into a unified KV-RAPTOR method to optimize both retrieval and generation performance. Unlike prior work that focuses either on enhancing retrieval structure to improve generation quality or solely on KV-cache reuse to improve performance, KV-RAPTOR jointly addresses both dimensions. This enables quality-preserving, low-latency generation under multilingual and variable Top-K conditions—an essential step toward scaling RAG pipelines for real-world applications.

## 3. KV-RAPTOR

This work proposes a scalable and inference-efficient extension of the RAPTOR pipeline that incorporates key-value (KV) cache compression and reuse techniques inspired by CacheGen. Our method enhances the standard RAG pipeline by supporting persistent token caches across recursive retrieval segments, enabling faster generation and reduced latency while preserving retrieval quality.

RAPTOR [Sarthi et al. 2024] is a RAG pipeline that constructs a hierarchical tree index through recursive abstraction of document chunks. Initially, input documents are segmented into smaller passages and embedded via a dense encoder such as SBERT [Reimers and Gurevych 2019]. These chunks are grouped using Gaussian Mixture Models (GMM) and recursively summarized using a language model (e.g., GPT-3.5-turbo), forming a tree in which parent nodes are increasingly abstract summaries of their children. This recursive structure supports two retrieval strategies: *tree traversal*, which selects relevant summaries layer by layer, and *collapsed tree*, which retrieves across the flattened tree. The use of soft clustering allows a chunk to belong to multiple clusters, better capturing overlapping semantics. The number of clusters is automatically selected via Bayesian Information Criterion (BIC), avoiding manual tuning. While RAPTOR is effective for hierarchical and semantically grounded retrieval, it does not address latency and inefficiencies during the decoding phase.

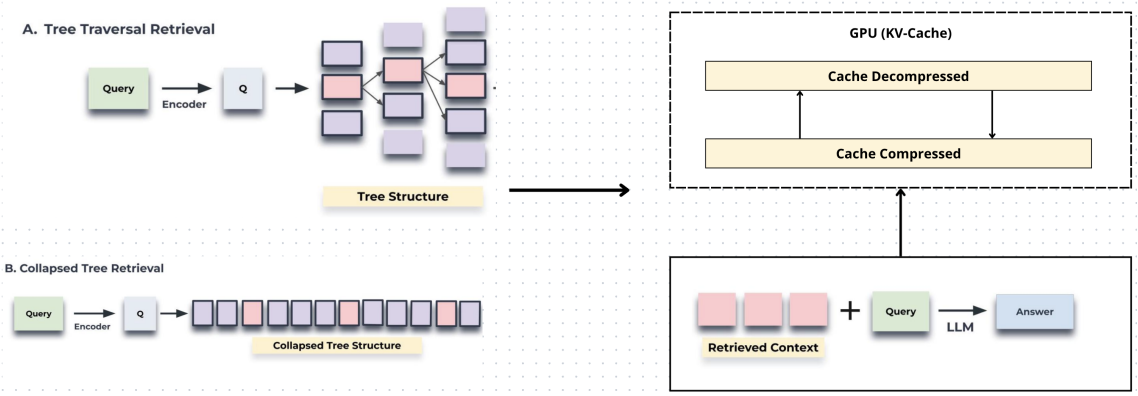



**Figure 1. KV-RAPTOR architecture: tree-based retrieval (left) from [Sarthi et al. 2024] combined with compressed KV-cache reuse for efficient generation (right). CacheGen [Liu et al. 2024b] modules operate entirely on GPU to reduce decoding latency.**

KV-RAPTOR extends RAPTOR by introducing compressed KV-cache reuse within the decoder, inspired by the design of CacheGen [Liu et al. 2024b]. In our method, retrieved segments are augmented with persistable KV-cache states that are compressed using distribution-aware quantization and encoded as compact streams. These caches are decoded on the GPU and reused across retrieval iterations to minimize redundant attention computations. Unlike standard prefix-based reuse mechanisms, our implementation supports partial recomputation of keys and values, allowing the model to flexibly integrate cache states even when the retrieved context is non-contiguous. Figure 1 illustrates the full KV-RAPTOR architecture.

To accelerate the RAPTOR indexing pipeline, we redesigned both the embedding and clustering stages to operate efficiently on GPUs. The original implementation computed sentence embeddings sequentially and relied on CPU-based UMAP for dimensionality reduction, resulting in substantial processing time for large-scale document collections. In our optimized approach, document passages are encoded in parallel batches on the GPU, which reduces kernel launch overhead and maximizes hardware utilization during embedding generation. Additionally, we replaced CPU-bound dimensionality reduction and clustering with GPU-parallel methods that retain all data in device memory. These leverage parallel graph construction and fuzzy set projections, as well as probabilistic clustering algorithms that compute responsibilities concurrently across threads [Nolet et al. 2021].

**Algorithm 1** Optimized GPU Indexing Pipeline for Tree-Based Retrieval

**Require:** Document collection $D = \{d_1, d_2, \ldots, d_n\}$
**for all** documents $d_i$ in $D$ **do**
  Segment $d_i$ into fixed-size passages $C_i$
**end for**
Merge all passages into global set $C$
Encode $C$ in parallel batches using GPU embedding model $\rightarrow E$
Retain $E$ in device memory for downstream operations
Construct k-NN graph and project $E$ to low-dimensional space using GPU-based UMAP
Perform GMM clustering with parallel responsibility computation $\rightarrow$ clusters $\{k_j\}$
**for all** clusters $k_j$ **do**
  Aggregate cluster content and generate parent summary with LLM
**end for**
**return** Recursive tree index with summarized nodes

This approach avoids expensive host-to-device memory transfers and benefits from high memory bandwidth and thread-level parallelism, resulting in a significant reduction in indexing time. The overall process is summarized in Algorithm 1, which outlines the parallelized workflow used to construct the hierarchical tree structure from raw document collections.

By tightly coupling recursive retrieval with cache-aware generation, KV-RAPTOR introduces a principled approach to reducing decoding latency while preserving the contextual richness of long retrieved documents in a scalable indexing process for large

databases. The following sections describe the experimental setup used to evaluate its effectiveness and analyze its impact on latency and quality across real-world multilingual datasets.

## 4. Material and Methods

This study investigates whether it is possible to accelerate retrieval-augmented generation (RAG) pipelines while preserving answer quality by integrating hierarchical retrieval structures with cache-based inference optimizations. To explore this, we reproduce the Recursive Abstractive Processing for Tree-Organized Retrieval framework from RAPTOR and implement a Naive RAG baseline for comparison. Both pipelines are evaluated in configurations with and without KV-Cache optimization via CacheGen. This setup allows for a systematic analysis of the trade-offs between retrieval structure, inference latency, and output quality, including how these factors behave across languages and under varying retrieval depths.

### 4.1. Datasets

We conduct experiments using two distinct datasets to evaluate our retrieval-augmented generation approaches: NarrativeQA and Pirá.

NarrativeQA dataset [Kočiský et al. 2018] consists of narrative-style documents in English, comprising 1,572 unique document summaries after filtering. It was selected due to its extensive use in evaluating generative models' ability to interpret and answer complex, narrative-driven questions, also is one of datasets used in the original paper.

Pirá dataset [Paschoal et al. 2021] is a multilingual corpus primarily containing documents in Portuguese, complemented by some in English. After preprocessing and filtering, 757 unique document summaries remained. This dataset was included to assess the effectiveness of our retrieval methods in handling linguistically diverse and morphologically rich text, specifically addressing challenges posed by the Portuguese language.

### 4.2. Environment

For the retrieval step, cosine similarity was employed as the search method due to its computational efficiency and effective representation of semantic similarity between document embeddings. All retrieval and generative tasks were conducted using the Meta Llama 3.1 8B Instruct model, an instruction-tuned LLM optimized for multilingual dialogue and text generation tasks.

To comprehensively assess the effectiveness of our proposed KV-RAPTOR optimized retrieval-augmented generation (RAG) pipeline, we divided evaluation metrics into two categories: quality metrics and performance metrics.

Quality was measured using standard natural language generation metrics widely adopted in the literature. We computed ROUGE-L [Lin 2004], which evaluates the longest common subsequence between generated and reference answers, BLEU-1 and BLEU-4 [Papineni et al. 2002], which assess precision at different n-gram levels, and METEOR [Lavie and Agarwal 2007], a metric designed to account for linguistic variations through stemming and synonymy. Sentence-level variants of BLEU were used to adapt to short-form answers.

Performance was evaluated based on system-level latency and resource efficiency. Specifically, we measured: (i) *Time-To-First-Token (TFFT)*, which captures the time interval from query submission to the first generated token; (ii) *End-to-End Latency (e2e_latency)*, defined as the total time from input to complete output; and (iii) *GPU Memory Usage*, which provides insights into runtime memory overhead. These performance metrics are aligned with benchmarking guidelines provided by NVIDIA for LLM deployments [NVIDIA Corporation 2024].

The experiments were executed on a cloud-based infrastructure provided by RunPod [RunPod 2025], featuring an NVIDIA A40 GPU with 48GB of VRAM, 9 virtual CPUs, and 50GB of RAM, at an approximate cost of $0.44 per hour.

## 5. Results and Discussion

In this section, we present an experimental study designed to evaluate the contributions of our proposed optimizations. We structure our analysis in two main parts: first, we present and validate the GPU-accelerated indexing strategy, which significantly reduces preprocessing time through parallel embedding and clustering operations. Then, we assess the complete KV-RAPTOR retrieval and generation pipeline by comparing it against a Naive RAG baseline and the original RAPTOR framework, focusing on key metrics such as latency, generation quality, and scalability under different retrieval depths. We also investigate how the proposed method performs across linguistically distinct datasets and under varying system constraints. This dual-layer evaluation enables a deeper understanding of how structured retrieval and inference-time caching can jointly optimize real-world RAG systems. The objective of our experiments is to answer the following research questions:

RQ1) Can CacheGen's KV-Cache optimization reduce latency without compromising the quality of the RAPTOR retrieval and generation pipeline?

RQ2) Does the RAPTOR tree-based retrieval structure provide quality gains when applied to Portuguese-language documents?

RQ3) How is latency affected in KV-RAPTOR as the retrieval context size (Top-K) increases?

**Indexing**. Prior to retrieval, both datasets were subjected to preprocessing and dense vector indexing using ChromaDB as the underlying vector database. Document embeddings were generated using language-specific encoders, and standard fixed-size chunking strategies were applied to ensure fine-grained retrieval.

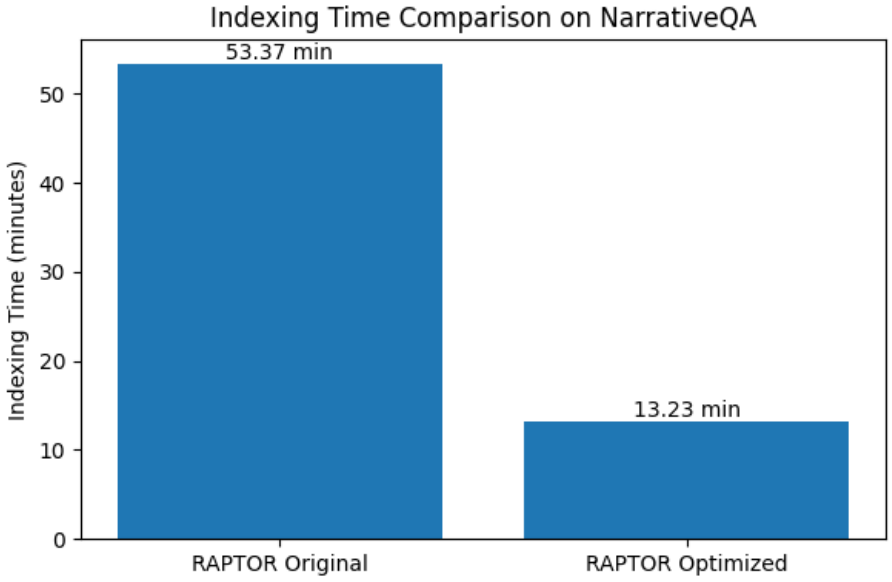



**Figure 2. Indexing time comparison on the NarrativeQA dataset. Optimizations in KV-RAPTOR reduced the total processing time by over 70%.**

For the NarrativeQA dataset, the encoder *multi-qa-mpnet-base-cos-v1* from the SBERT family [Reimers and Gurevych 2019] was used, replicating the original RAPTOR configuration. The baseline flat retrieval configuration indexed a total of 14,738 document chunks, while RAPTOR indexed 15,825 chunks through recursive abstraction and hierarchical clustering. During the summarization stage of RAPTOR's indexing, we employed OpenAI's GPT-3.5-turbo model, consistent with the original methodology.

Indexing time was significantly reduced through the optimizations described in Section 3, which introduced batched GPU-based embedding and clustering via parallel device-resident operations. These changes enabled a substantial reduction in total indexing time. On the NarrativeQA dataset, the original RAPTOR indexing procedure required approximately 44 minutes and 36 seconds, while our optimized version reduced this to 13 minutes and 14 seconds. Figure 2 illustrates this performance improvement, highlighting a reduction of over 70% in indexing time.

For the Pirá dataset, embedding generation was performed using the BERTimbau Base [Souza et al. 2020] model, released as *neuralmind/bert-base-portuguese-cased*, which is pre-trained specifically for Brazilian Portuguese. The baseline retrieval setup produced 5,260 chunks, while the RAPTOR variant indexed 5,645. As in the English dataset, GPT-3.5-turbo was used to generate intermediate summaries during tree construction.

**Retrieval and Generation**. For retrieval and generation tests in both datasets, NarrativeQA and Pirá, we sampled a subset of 10 queries, each executed 10 times, totaling 100 runs per experiment. This repetition is necessary to demonstrate the impact of CacheGen, which benefits from token reuse in high-frequency generations enabled by the KV-Cache mechanism. We also include a baseline using Naive RAG to validate the effectiveness of RAPTOR's tree-based retrieval and to evaluate overall improvements in quality and latency.

We begin with results from the NarrativeQA English dataset. As shown in Table 1, KV-RAPTOR, achieves ROUGE-L of 43.89, BLEU-1 of 34.93, BLEU-4 of 15.07, and METEOR of 33.31—closely aligning with RAPTOR's original scores of 46.82, 35.71, 15.69, and 34.88, respectively. These results demonstrate that the KV-Cache optimization introduced by CacheGen does not substantially compromise output quality, while still benefiting from the retrieval improvements provided by RAPTOR's tree-based structure.

Regarding latency, our implementation of KV-RAPTOR reduces the time-to-first token (TFFT) and end-to-end latency, from 0.15 and 0.16 seconds (Naive RAG and RAPTOR) to 0.06 seconds in the optimized method. This improvement is achieved with increased GPU memory usage: from 34.61% without cache to 83.58% with CacheGen. The results indicate that CacheGen improves performance by leveraging KV-Cache, while maintaining comparable quality levels.

**Table 1. Quality and performance comparison on NarrativeQA (Top-10 retrieval). KV-RAPTOR reduces TFFT by over 50% with minimal quality impact.**

| Model | Quality | | | | Performance | | |
|---|---|---|---|---|---|---|---|
| | ROUGE-L | BLEU-1 | BLEU-4 | METEOR | TFFT | e2e_Latency | GPU Mem. |
| Naive RAG | 37.12 | 29.06 | 8.18 | 26.67 | 0.15s | 0.41s | 34.61% |
| RAPTOR | 46.82 | 35.71 | 15.69 | 34.88 | 0.16s | 0.41s | 34.61% |
| KV-RAPTOR | 43.89 | 34.93 | 15.07 | 33.31 | 0.06s | 0.32s | 83.58% |

Following the previous experiment, increase the retrieval context from Top-10 to Top-20 documents does not lead to significant degradation in quality. As shown in Table 2, all four quality metrics (ROUGE-L, BLEU-1, BLEU-4, and METEOR) exhibit only minor variations across models compared to the Top-10 setup. For instance, KV-RAPTOR achieves BLEU-4 of 14.46 and METEOR of 33.13— close to the 15.07 and 33.31 scores observed previously—while also slightly improving BLEU-1 to 32.87.

These results indicate that expanding the retrieval scope to 20 documents provides a richer context with only minimal impact on generation quality. This reinforces the scalability of RAPTOR's retrieval structure and demonstrates that CacheGen continues to preserve quality under larger Top-K configurations.

**Table 2. NarrativeQA results with Top-20 retrieval. KV-RAPTOR achieves 75% TFFT reduction under increased context, maintaining comparable quality.**

| Model | Quality | | | | Performance | | |
|---|---|---|---|---|---|---|---|
| | ROUGE-L | BLEU-1 | BLEU-4 | METEOR | TFFT | e2e_Latency | GPU Mem. |
| Naive RAG | 36.60 | 27.02 | 7.93 | 26.15 | 0.28s | 0.56s | 34.61% |
| RAPTOR | 43.15 | 31.33 | 13.78 | 29.80 | 0.29s | 0.54s | 34.61% |
| KV-RAPTOR | 42.64 | 32.87 | 14.46 | 33.13 | 0.07s | 0.35s | 83.70% |

From a performance perspective, the KV-Cache optimization continues to show effectiveness. TFFT remains low in the KV-RAPTOR, increasing marginally from 0.06s to 0.07s under the expanded context. However, end-to-end latency increases across all models. This is expected, as CacheGen accelerates only the generation layer and the cost

of retrieving more documents impacts total latency. Nevertheless, this trade-off remains acceptable, as the retrieval time increase leads to higher context quality, while generation speed is preserved.

Given our first research question (RQ1) objective to reduce latency without sacrificing retrieval and generation quality, we adopt Top-20 as the standard configuration in subsequent experiments. This setting provides a favorable balance between enriched context and generation latency efficiency.

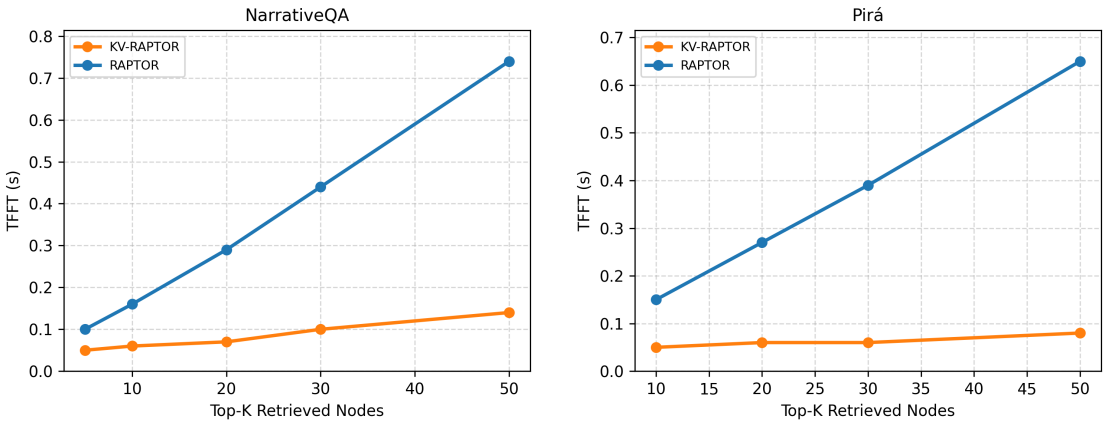
Table 3 presents the results for the Pirá dataset under a Top-20 retrieval configuration. KV-RAPTOR outperforms both Naive RAG and the original RAPTOR variant across all quality metrics, while also significantly reducing TFFT and end-to-end latency, suggesting that cache reuse strategies may provide greater benefit in linguistically complex or morphologically rich languages such as Portuguese. These findings support Research Question 2 (RQ2), demonstrating that tree-structured retrieval paired with cache-aware generation maintains quality while improving efficiency in non-English settings.

**Table 3. Pirá Portuguese dataset results with Top-20 retrieval. KV-RAPTOR shows improved quality and lowest latency. KV-RAPTOR surpasses the original RAPTOR, suggesting that cache reuse may offer greater gains in morphologically rich languages.**

| Model | Quality | | | | Performance | | |
|---|---|---|---|---|---|---|---|
| | ROUGE-L | BLEU-1 | BLEU-4 | METEOR | TFFT | e2e_Latency | GPU Mem. |
| Naive RAG | 21.15 | 20.25 | 7.58 | 22.26 | 0.27s | 1.34s | 34.69% |
| RAPTOR | 21.65 | 20.81 | 9.28 | 23.34 | 0.27s | 1.30s | 34.69% |
| KV-RAPTOR | 23.18 | 24.31 | 11.32 | 25.91 | 0.06s | 0.90s | 83.92% |

The narrow performance gap between Naive RAG and RAPTOR in this dataset may reflect challenges in summarization quality during tree construction or encoder-context alignment, even when using a domain-specific model for Portuguese input. We also observe that average response length in the Pirá dataset is 29 tokens, compared to just 9 tokens for NarrativeQA. This variation reflects the nature of the datasets, where Pirá includes broader discursive content, while NarrativeQA focuses on shorter, often noun-based answers.

To conclude the experimental analysis, Figure 3 addresses Research Question 3 (RQ3), examining the impact of retrieval scope on latency. As the number of retrieved documents (Top-K) increases, KV-RAPTOR maintains near-constant TFFT, whereas the original RAPTOR exhibits a linear increase. This demonstrates the efficiency of token reuse via KV-Cache, even under scaling conditions.




**(a)** NarrativeQA **(b)** Pirá

**Figure 3. Effect of Top-K size on TFFT for NarrativeQA and Pirá datasets. CacheGen-optimized maintains low latency as retrieval scope increases.**

From a systems perspective, these results highlight the feasibility of integrating tree-based retrieval structures within generative pipelines for large and heterogeneous document collections. KV-RAPTOR shows promise for practical deployment in retrieval-augmented generation over semi-structured or document-oriented databases by enabling a favorable balance between latency and quality. This performance–scalability trade-off, achieved through structured retrieval and caching mechanisms, forms the basis of our concluding discussion in the next section.

## 6. Conclusion and Future Work

This study introduced KV-RAPTOR, a novel method that extends the RAPTOR architecture by embedding compressed key-value (KV) cache reuse mechanisms inspired by CacheGen into its recursive, tree-based retrieval pipeline. This method enables substantial reductions in inference latency while preserving the contextual relevance and generation quality expected from structured retrieval.

Our experimental results on multilingual datasets, including a Portuguese-specific corpus, demonstrate that KV-RAPTOR achieves up to a 75% reduction in time-to-first-token latency, with minimal impact on output quality. Additionally, indexing optimizations such as batched embedding generation and GPU-accelerated clustering reduced preprocessing time by over 70%, reinforcing the method's scalability for large and diverse document collections.

Finally, this research contributes with a combined evaluation methodology that articulates generation quality metrics with computational performance indicators. This approach allows a more complete analysis of the trade-offs between quality and performance, establishing guidelines for developing more efficient, robust RAG pipelines applicable to real production scenarios.

**Limitations.** Our evaluation primarily focused on RAPTOR as the underlying retrieval architecture, extending the experiments to other tree-based RAG structures, such as HippoRAG [Jimenez Gutierrez et al. 2024] or GraphRAG [Edge et al. 2025], could pro-

vide broader insights into generalizability. Additionally, the experiments were limited to Meta Llama 3.1 8B Instruct model.

The current proposal assumes that segments retrieved and generated in previous stages of the pipeline will be reused in subsequent iterations. However, the absence of public benchmarks containing realistic context reuse logs limits more accurate modeling of this behavior. This gap represents a strategic opportunity: building or adapting datasets that simulate actual query patterns could allow the development of more robust adaptive caching strategies.

Future work will explore extending KV-RAPTOR to support dynamic indexing updates, adaptive retrieval depths, and more granular cache reuse across intermediate pipeline stages. We also intend to evaluate this method applied to other hierarchical retrieval structures and test compatibility with other LLM architectures.

## 7. Acknowledgements

This work has been funded by P&D CEMIG/ANEEL PD-04950- D0677/2023 supported by the Advanced Knowledge Center in Immersive Technologies (AKCIT), with financial resources from the PPI IoT/Manufatura 4.0 / PPI HardwareBR of the MCTI grant number 057/2023, signed with EMBRAPII. This work was also supported by the National Institute of Science and Technology (INCT) in Responsible Artificial Intelligence for Computational Linguistics and Information Treatment and Dissemination (TILD-IAR) grant number 408490/2024-1.

## Reproducibility Statement

All code, data processing scripts, and experimental configurations used in this study are available at: `https://github.com/joaogabrieljunq/kv-raptor`

## Environmental Footprint Statement

All experiments were conducted on a cloud-based platform using a single NVIDIA A40 GPU, utilized exclusively for GPU-accelerated processing. We were not able to precisely estimate the total energy consumption for this study. However, we recognize the importance of energy efficiency and plan to incorporate sustainability-aware benchmarks in future research.

## References

Aquino, I., dos Santos, M. M., Dorneles, C., and Carvalho, J. T. (2024). Extracting information from brazilian legal documents with retrieval augmented generation. In *Anais Estendidos do XXXIX Simpósio Brasileiro de Bancos de Dados*, pages 280–287, Porto Alegre, RS, Brasil. SBC.

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Metropolitansky, D., Ness, R. O., and Larson, J. (2025). From local to global: A graph rag approach to query-focused summarization.

Jiang, C., Gao, L., Zarch, H. E., and Annavaram, M. (2024). Efficient llm inference with i/o-aware partial kv cache recomputation.

Jimenez Gutierrez, B., Shu, Y., Gu, Y., Yasunaga, M., and Su, Y. (2024). Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems*, 37:59532–59569.

Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. (2018). The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Lavie, A. and Agarwal, A. (2007). Meteor: an automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, page 228–231, USA. Association for Computational Linguistics.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Li, B., Jiang, Y., Gadepally, V., and Tiwari, D. (2024). Llm inference serving: Survey of recent advances and opportunities. In *2024 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8.

Li, H., Li, Y., Tian, A., Tang, T., Xu, Z., Chen, X., Hu, N., Dong, W., Li, Q., and Chen, L. (2025). A survey on large language model acceleration based on kv cache management.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Liu, A., Liu, J., Pan, Z., He, Y., Haffari, G., and Zhuang, B. (2024a). Minicache: Kv cache compression in depth dimension for large language models. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C., editors, *Advances in Neural Information Processing Systems*, volume 37, pages 139997–140031. Curran Associates, Inc.

Liu, Y., Li, H., Cheng, Y., Ray, S., Huang, Y., Zhang, Q., Du, K., Yao, J., Lu, S., Ananthanarayanan, G., Maire, M., Hoffmann, H., Holtzman, A., and Jiang, J. (2024b). Cachegen: Kv cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, page 38–56, New York, NY, USA. Association for Computing Machinery.

Nolet, C. J., Lafargue, V., Raff, E., Nanditale, T., Oates, T., Zedlewski, J., and Patterson, J. (2021). Bringing umap closer to the speed of light with gpu acceleration.

NVIDIA Corporation (2024). Benchmarking metrics for large language models. `https://docs.nvidia.com/nim/benchmarking/llm/latest/metrics.html`. Accessed: 2025-04-17.

Oliveira, V. P. L. (2024). *MemoryGraph: uma proposta de memória para agentes conversacionais utilizando grafo de conhecimento*. Tese (doutorado em ciência da computação), Universidade Federal de Goiás, Goiânia.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Paschoal, A. F. A., Pirozelli, P., Freire, V., Delgado, K. V., Peres, S. M., José, M. M., Nakasato, F., Oliveira, A. S., Brandão, A. A. F., Costa, A. H. R., and Cozman, F. G. (2021). Pirá: A bilingual portuguese-english dataset for question-answering about the ocean. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 4544–4553, New York, NY, USA. Association for Computing Machinery.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

RunPod (2025). Runpod – cloud compute for ai, ml, and more. Acesso em: 28 abr. 2025.

Sarthi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., and Manning, C. D. (2024). RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

Souza, F., Nogueira, R., and Lotufo, R. (2020). BERTimbau: pretrained BERT models for Brazilian Portuguese. In *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*.

Taschetto, L. and Fileto, R. (2024). Using retrieval-augmented generation to improve performance of large language models on the brazilian university admission exam. In *Anais do XXXIX Simpósio Brasileiro de Bancos de Dados*, pages 799–805, Porto Alegre, RS, Brasil. SBC.

Yao, J., Li, H., Liu, Y., Ray, S., Cheng, Y., Zhang, Q., Du, K., Lu, S., and Jiang, J. (2025). Cacheblend: Fast large language model serving for rag with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, page 94–109, New York, NY, USA. Association for Computing Machinery.

Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., and Liu, Z. (2024). Evaluation of retrieval-augmented generation: A survey.