Expanding Horizons in RAG: Exploring and Extending the Limits of RAPTOR

Stanford CS224N Custom Project

Alex Laitenberger
Department of Computer Science
Stanford University
alaiten@stanford.edu

Abstract

"Retrieval-Augmentation Generation" (RAG) systems have emerged in response to the current constraints faced by large language models (LLMs) when applied to domain-specific tasks, as discussed by Lewis et al. (2021). These systems harness the power of LLMs to perform complex reasoning across tightly focused document sets, proving particularly valuable in specialized research areas or within specific organizational contexts, such as company business processes that rely on proprietary or department-specific documentation. At the International Conference on Learning Representations (ICLR) Sarthi et al. (2024) introduced "Recursive Abstractive Processing for Tree-Organized Retrieval" (RAPTOR), an innovative strategy designed to address the inherent limitations of RAG systems. The RAPTOR framework is fundamentally segmented into two main components: 1) retrieval and 2) construction of the underlying retrieval tree system. In this project, I focused on critically evaluating and enhancing the construction component of RAPTOR. Through my investigation, I identified several key limitations of the existing framework and developed a new simplified approach that completely redesigns the construction of retrieval trees. This new methodology not only consistently surpassed the original system in handling both abstractive and extractive queries but also facilitated the development of more profound retrieval tree structures, enhancing the system's capacity to interpret and utilize different segments of documents effectively.

1 Key Information

- TA mentor: Anna Goldie
- External collaborators or mentor: No, Sharing project: No

2 Introduction

RAG systems have been developed to address the constraints faced by LLMs when dealing with domain-specific queries. These systems enhance LLM capabilities by augmenting prompts with domain-specific contexts, initially retrieved from a dedicated information retrieval system (Lewis et al., 2021). Despite the innovations in retrieval technology, current methods exhibit a significant limitation: they only process contiguous chunks of text, thereby failing to grasp the complete semantic depth of documents.

To address these challenges, the innovative RAPTOR, as proposed by Sarthi et al. (2024), aims to transform how information is processed and structured. RAPTOR enhances document understanding by recursively clustering related text chunks and summarizing them. This approach enables the construction of a hierarchical tree from the bottom up, effectively capturing both the meaning and the structural hierarchy of a document across various levels of abstraction. By integrating this structured

data into RAG systems, LLMs can access a richer context within prompts, potentially leading to more nuanced and contextually aware responses.

Despite its potential, RAPTOR exhibits certain limitations that can hinder its effectiveness in practical applications. One notable issue is the tendency to produce relatively flat tree structures, which may not adequately represent the complexity of the information within large documents. This is partly due to its reliance on Gaussian Mixture Models (GMMs) as the primary clustering algorithm, which can be suboptimal for managing high-dimensional text chunks. Furthermore, the expected benefits of soft clustering, which could theoretically enhance the granularity of document processing, are not fully realized in handling typical document sizes found in narratives or comprehensive NLP research papers. This shortfall diminishes the anticipated advantages of employing GMMs. Additionally, the current implementations of RAPTOR have shown a suboptimal approach to sentence tokenization, which could further compromise the system's efficacy in nuanced text parsing and retrieval tasks.

Building on the foundational ideas of RAPTOR, my approach involves a completely redesigned and simplified method for constructing the retrieval tree. It leverages the inherent tree structure generated by agglomerative clustering. Moreover, by incorporating positional information of text chunks, my method enhances insight into the underlying structure of documents, which is crucial for the clustering algorithm to effectively respect and utilize. The reimagined architecture of the retrieval tree through my approach has resulted in deeper tree structures that provide more levels of abstractive summaries. Consequently, the answers generated from these structures exhibit a profound understanding of various document sections, enriching the interpretative depth significantly. In comparative evaluations, my new approach consistently outperforms the original across both abstractive and extractive queries. Furthermore, the streamlined nature of this new approach renders the clustering component of tree construction computationally negligible.

3 Related Work

In the realm of RAG systems, foundational work by Lewis et al. (2021) and Akyurek et al. (2022) has explained the motivations behind retrieval augmentation, particularly highlighting the limitations of LLMs. This body of work has set the stage for my project, which further explores and extends the capabilities of RAG systems. Sarthi et al. (2024) introduced the innovative RAPTOR approach as a solution to overcome the limitations identified in previous RAG systems. I have chosen this particular study as the foundational paper for my project, aiming to delve deeper into its limitations and potential enhancements. Further advancements in RAG systems include the "RAG-end2end" proposal by Siriwardhana et al. (2023), which aims to create a more dynamic RAG system capable of adapting to domain-specific knowledge bases by updating all components of the external knowledge base during training. This proposal aligns closely with the objectives of my project, which seeks to refine the adaptability and efficiency of RAG systems.

Additionally, the introduction of rotational positional embeddings "Roformer" applied on transformer models by Su et al. (2023) has inspired the integration of positional embeddings within the context of RAPTOR trees in my project, aiming to enhance the structural and contextual understanding of document clusters. The application of sentence embeddings in my project is heavily influenced by the work of Reimers and Gurevych (2019), who introduced Sentence-bert (SBERT). This paper has been instrumental in providing the necessary insights for generating improvements in the clustering algorithms used in the RAPTOR system. Lastly, the QASPER dataset and evaluation script provided by Dasigi et al. (2021) are utilized in my project to assess the efficacy of the proposed improvements, ensuring that the enhancements are not only theoretical but also practical and measurable.

4 Approach

To find possible improvements for the RAPTOR clustering algorithm I start by taking a closer look at the existing implementation.

Chunking Text: The first step is chunking the text of a document into continuous chunks of 100 tokens. To prevent chunk splits within sentences, full sentences are exclusively used for chunking. Examining the official implementation reveals that the current method of text chunking, which relies on predefined delimiters and regular expressions, often splits sentences inappropriately, disrupting contextual relationships. Consequently, I have substituted this approach with the sentence tokenizer

from the Natural Language Toolkit (NLTK)[1], which produces cleaner and more contextually coherent text chunks.

Generating Embeddings: At the next step RAPTOR generates embeddings of the text chunks using the SBERT model from Reimers and Gurevych (2019). SBERT is a modification of BERT, that can generate semantically meaningful sentence embeddings, ideal for comparisons with cosine-similarity. For each text chunk the output is a single high-dimensional embedding, representing its full meaning.

Clustering: To cluster the resulting embeddings RAPTOR applies GMMs. They assume that data points are generated from a mixture of several Gaussian distributions. This way a data point can be "softly" clustered into multiple clusters, which seems to better represent the contextual nature of text, where sentences can be relevant to several topics. An observed challenge in the RAPTOR paper is the high dimensionality of the embeddings for GMMs, creating the need for dimensionality reduction. A possible down-side is the loss of information from the original SBERT embeddings. Additionally, as mentioned in the RAPTOR paper, GMMs may not perfectly align with the nature of text data, which follows a more sparse and skewed distribution. However, their empricial findings show that GMMs still are applicable. (Sarthi et al., 2024)

Summarization: In the next step the clustered text chunks are summarized using a LLM with a summarization prompt and the text chunks within a cluster as context. With the resulting output text another embedding is created with SBERT. The new text and embedding is then used to build the tree structure.

Tree Building: With all previous steps in place the tree building process of RAPTOR begins. It starts with the leaf nodes containing the initial text chunks and their embeddings. The next upper level consists of their parent nodes according to the clustering result, containing the summarized texts and their embeddings. The process of clustering, summarizing and tree building is repeated until no more upper layers can be created.

Alternative Summarization Approach: Looking at this process it caught my attention, that the task of generating embeddings within SBERT is actually very similar to the task of generating summaries over text chunks with an LLM. I therefore propose to examine the possibility of creating the summaries directly with the embeddings of the text chunks, using a similar approach as SBERT. This would avoid the extra step and possible loss that comes with summarizing on text chunks using a summary prompt and might even lead to a reduction of the reported 4% hallucination rate in RAPTOR summaries in Sarthi et al. (2024).

Adding Positional Information: One other notable aspect of the RAPTOR process, particularly in terms of text chunking and clustering, is that the clustering does not respect any positional information of the text chunk it clusters. Assuming that near text chunks generally have a higher probability of belonging to the same context I propose to capture positional information of text chunks within their embeddings, so they can be respected during clustering. This could be implemented by 1) using absolute positional information and encode it in the embeddings or 2) using relative position embeddings, such as rotational position embeddings that were applied to transformers in Su et al. (2023). Using an alternative clustering algorithm to GMMs that would avoid dimensionality reduction could also help to maintain this added positional information for clustering.

Visualizing the tree: To get a better intuition of the resulting tree I present a tree visualization in Figure 1.

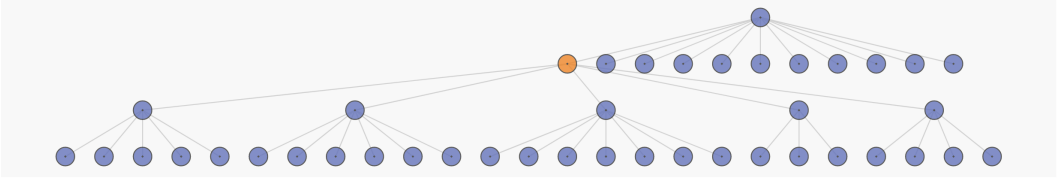

Figure 1: RAPTOR tree for Cinderella story.

[1] `https://www.nltk.org/api/nltk.tokenize.html`

The visualization has been realized based upon the provided code of a pull request, that adds tree visualization capabilities to the official RAPTOR implementation[2] I added a slight modification to differ between the actual RAPTOR nodes (in blue) and an artifically created top node (orange) for visualization purposes. It shows the structure of a generated RAPTOR tree for the Cinderella story, which is used as a simple demo text. We can observe, that the result is actually not a single tree, but six distinct trees. And the overall structure is generally very flat. It consists of six root nodes containing all the leaf nodes. In order to better support the retrieval I suggest to create a final single root node on top of the sub-trees, containing a summary over all content. Besides that, the trees do not show any soft clustering applied in the way, that a leaf node would have been assigned to multiple parent nodes. As a response to a review question on soft clustering the authors of the RAPTOR paper commented, that they observed actual soft clustering with documents over 14,300 tokens[3]. The Cinderella narrative contains about 2,600 tokens and a sample of an NLP paper from the QASPER dataset contains about 6000 tokens. This negates the originally proposed advantage of GMMs to be able to perform soft clustering for these kinds of documents. And in terms of wording, if we were to envision the structure with applied soft clustering, it would actually no longer be a tree, but a directed acyclic graph (DAG), resembling an interconnected web more than a tree.

Visualizing Embeddings and Clustering: To get a an additional intuition for the way the embeddings are clustered with GMMs, I present a visualization of a sample clustering result in figure 3 in Appendix A. In this 2D representation it appears, that clusters are formed in various shapes across the embeddings, which was a highlighted advantage of GMMs. Moreover, very discrete cluster boundaries can be observed, with a few exceptions, suggesting that soft clustering may not be effectively applied.

Finding Clustering Alternatives: To identify possible alternatives for GMMs as the clustering algorithm I have conducted several discussions with the authors of the RAPTOR paper and worked through the comments of the reviewers on Open Review. One reviewer specifically suggests to try agglomerative clustering as an alternative.3 Additionally, scikit-learn.org provides a comprehensive overview on clustering algorithms.[4] After reviewing different options, I suggest considering several alternatives for experimentation, including: 1) agglomerative clustering, 2) a neural clustering approach where a trained model can perform the clustering task. A neural approach seems to be especially promising considering the high dimensionality of text embeddings.

Collecting Suggestions and Scope: Table 1 collects all identified suggestions from the upper sections for improving RAPTOR, that resulted from my analysis. As a scope for this project I select to implement and experiment on agglomerative clustering, NLTK sentence tokenizer, positional embeddings and adding a single root node at the top of the tree.

Agglomerative clustering is a hierarchical clustering method in which each input element initially represents its own cluster. These clusters are then progressively merged according to a specified distance criterion until all elements are contained within a single cluster. The steps executed during this process are recorded in what is known as a linkage matrix, which can be visualized in a dendrogram (see figure 4). This methodology allows for a detailed examination of relationships between data points and clusters at different levels of abstraction.[5]

For configuration, the agglomerative clustering library supports various linkage criteria. I opted for the average method, which computes the average distance between elements in one cluster and a merge candidate at each step. This method is particularly suitable for sentence embeddings. Given that SBERT vectors are optimized for cosine similarity, this measure is utilized to determine distances in agglomerative clustering. This approach effectively leverages the inherent properties of SBERT vectors to capture semantic similarities between text chunks.

Examining the dendrogram shown in Appendix B figure 4, I observed that the resulting structure already represents a complete tree, making it seem like an ideal candidate to construct hierarchical summaries such as intended in the RAPTOR paper. Moreover the linkage matrix already contains all information for subsequent clusters from bottom up. In contrast to the original RAPTOR approach utilizing GMMs, the clustering in this case does not need to be applied iteratively to determine

[2] https://github.com/parthsarthi03/raptor/pull/40

[3] https://openreview.net/forum?id=GN921JHCRw

[4] https://scikit-learn.org/stable/modules/clustering.html

[5] https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering

Area	Suggestions
Alternatives to GMMs as base clustering algorithm	• Agglomerative • Neural
Chunking Text, Embeddings	• NLTK sentence tokenizer • Integrate positional embeddings – absolute – relative (e.g. rotary)
Summarization and tree building	• Merge sentence embeddings using a model similar to SBERT • Add a single root node to the tree with a full document summary

Table 1: Suggestions for Improving RAPTOR Clustering Algorithm

subsequent clusters. In the redesign of the clustering process, I have simplified each step by eliminating the need for iterating over nodes to check sizes and perform reclustering. Additionally, the original two-step clustering approach, which transitioned from broad to detailed clustering, and the need to repeatedly apply GMMs to determine the optimal number of clusters (e.g. up to 50 iterations), have been removed. This streamlining significantly simplifies the entire process. In my observations of the tree-building process with document sizes around 6000 tokens, such as those found in NLP research papers, the clustering component was practically negligible. Instead, the process is predominantly influenced by the round-trip times of LLM prompts for QA and summaries. Additionally, dimensionality reduction is no longer necessary, making clustering directly on full SBERT embeddings more meaningful. However, the replacement of GMMs with agglomerative clustering necessitates a full redesign of the tree-building process, as the clustering algorithm cannot simply be replaced.

New tree building: To avoid flat and loose structures as observed in the current RAPTOR trees, I propose a tree structure with a total of four levels: the leaf layer, two intermediate layers, and a single top root node at the top of the tree. To establish the intermediate layers, I cut the tree structure in the dendrogram at two predefined positions (with n being the number of leaf nodes): 1) at n/3 merge steps and 2) at n/6 merge steps, counted from the last merge at the top of the dendrogram. This segmentation strategy is anticipated to result in n/3 and n/6 clusters at these respective points, enhancing the structural depth and organization of the tree. The choice of these specific cuts was determined by examining the dendrogram and through several trials examining resulting tree visualizations, appearing sensible for an initial design. However, it may be beneficial to reconsider these values in the future and to explore experimentally various configurations. This could include a dynamic approach that allows for more levels in the tree, potentially yielding a more refined and adaptive clustering structure. The linkage matrix, that results from applying agglomerative clustering, can be used to extract clusters at specified distances, which is pictorially like cutting the tree structure in the dendrogram at a certain position on the y-axis. For implementation I use the hierarchical clustering library from scipy.[6] Finally, the top root node is realized by summarizing the texts of the nodes of the upper intermediate layer to get one condensed comprehensive summary over the whole document within a certain token threshold, which can serve as a short high-level overview for RAG prompts. An example for the new resulting tree can be seen in figure 2.

Positional embedings: To conduct initial experiments with the idea of positional embeddings I start by extending the SBERT embeddings of the text chunks with elements containing absolute positional information. To realize this I create three elements for each text chunk containing 1) its index 2) its

[6] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.fcluster.html`

distance to the end of the document, 3) the section it is contained in, assuming three equally sized sections in a document. These positional elements are normalized and scaled to be in the range of -0.1 to 0.1, so that they match the range of the SBERT embeddings and do not undermine their effects. In the final step, I introduce a priority factor that multiplies the magnitude of the positional elements. This adjustment enables experimentation with the extent to which positional information is integrated with the semantic information of the SBERT embeddings for the calculation of cosine similarity. With a few tests and qualitative observations of resulting tree structures I found a priority factor of 5.0 to be a good balanced starting point for quantitative experiments. The extended embeddings are only used for the agglomerative clustering process. They are not used for retrieval, which is still using the original SBERT embeddings to match against queries.

Implementation: To implement my proposals I have used the official RAPTOR implementation[7] and made substantial changes to the cluster-tree-builder class and rewrote the whole RAPTOR-clustering class to integrate with the new tree builder and to use agglomerative clustering. Additionally I have adjusted the split-text function in the utils class to use NLTK for sentence tokenization as descibed in the upper section "Chunking Text".

Baseline and Experiments: To set a baseline, I start by running the RAPTOR system as it is currently published on Github, evaluate the current state with the official QASPER dataset in Dasigi et al. (2021) and compare the results to those published in the RAPTOR paper. I use Google's Gemini 1.0 Pro as the language model to perform the baseline test, to contribute new experimental data. Therefore, I extend the current RAPTOR implementation to use Gemini as the language model for text summarization and QA. A challenge with this extension are the safety blocks of Gemini. As a modification I adjust the safety filters to block as little as possible, but 9 out of 416 papers cannot be summarized and have to be skipped for the experiment. A challenge with reproducing the original RAPTOR paper's QASPER experiments is to determine the specific prompting method for the QA part, that satisfies the QASPER evaluation expectations for the answers and their four different answer types. To perform the baseline test I write a Jupyter Notebook to extract the evaluation data from the QASPER dataset, run it against the RAPTOR implementation and collect the results of the evaluation questions. As a final step I use the official QASPER evaluation script to calculate the resulting quantitative metric.

5 Experiments

5.1 Data

Data: QASPER contains 5,049 questions across 1,585 NLP papers divided into three sets. The test set contains 416 papers. Each paper dataset contains questions and reference answers. QASPER expects a model's answer to match one of four different answer types: 1) Extractive, 2) Abstractive, 3) Boolean (yes/no), 4) None ("Unanswerable") (Dasigi et al., 2021).

5.2 Evaluation method

The accuracy of the QASPER evaluation is measured using an Answer-F1 metric. It is reported as a general Answer-F1 over all answers and for each answer type individually.

5.3 Experimental details

For the QASPER evaluation I use Gemini 1.0 Pro as the language model for text summarization and QA prompts and SBERT from Reimers and Gurevych (2019) as the model for creating the embeddings for text chunks. With these prepared I generate RAPTOR trees for the papers contained in the QASPER test set. In a next step I generate answers to the given questions in the test set, using a prepared prompt. With the generated answers I run the official QASPER evaluation script to compute the Answer-F1 score and the scores for individual answer types. The full experiment over all 416 papers in the test set takes several hours to complete on a local machine (Macbook M1 Pro 2021).

Metric	Baseline	New tree	POS 5.0	POS 5.0 (II)
Answer F1	39.79%	38.07%	39.26%	36.01%
Answer F1 by Type				
Extractive	37.37%	39.17%	**40.19%**	37.53%
Abstractive	17.48%	18.86%	18.33%	**19.11%**
Boolean	**56.15%**	40.89%	42.62%	40.68%
None	**85.36%**	76.16%	80.88%	68.52%
Missing Predictions	64	31	34	42

Table 2: QASPER Results for RAPTOR with SBERT and Gemini 1.0 Pro

5.4 Results

Table 2 presents the Answer-F1 metric as 39.79% for the baseline experiment. This result falls between the 36.70% achieved using UnifiedQA-3B and the 53.1% obtained with GPT-3.5 as the language model, as reported in the original RAPTOR paper by Sarthi et al. (2024). The results for the new tree approach show distinct improvements for extractive **(+1.8%)** and abstractive **(+1.38%)** questions. At the same time, the results for the answer types boolean and none declined significantly, leading to a worse total Answer F1 (-1.72%). The next experiment including positional embeddings into the new tree approach showed further improvements for extractive questions **(+1.02%)**, a decrease on abstractive questions (-0.53%) and a better performance on boolean and none-type questions leading to a better total Answer F1 **(+1.19%)**. The enhanced performance observed for both extractive and abstractive questions for the new tree approach aligned with initial expectations and qualitative assessments. However, the substantial decline in performance for boolean and none-type questions was unexpected. This performance is contingent upon several factors, including the right interpretation of the LLM of the expected answer type and format. To find the precise causes of these discrepancies, further analyses and experiments are necessary. It is interesting to note that the integration of positional embeddings with a priority factor of 5.0 further enhanced the results obtained from the new tree approach. The count of missing predictions indicates the number of questions that remained unanswered due to the Gemini blocks previously described. The QASPER test set comprises a total of 1,451 questions. A secondary test using the new tree structure and positional embeddings showed significant outcome variations, particularly in the 'none' category, with a 2-3% deviation in the extractive category questioning the QASPER Answer F1 metric's reliability for my experiments. Despite deterministic clustering and text chunking operations, and setting the QA and summarization prompts' temperature on Gemini 1.0 Pro to 0.0 for reproducibility, inconsistencies in Gemini's response behavior suggest potential causes for these deviations. Further investigation is needed to confirm this. Nonetheless, all tests with the new setup outperformed the original RAPTOR method in F1-score for extractive and abstractive responses, indicating potential improvements, pending validation with a more stable setup and possibly more insightful metrics.

6 Analysis

To analyse the effects of the new tree building approach let us take a look at the newly constructed retrieval tree for the Cinderella narrative in figure 2 and compare it to the original one from figure 1. As before the orange colored node is an artifically created node for visualization purposes. In the new tree figure it directly leads to the newly created top root node, which contains a comprehensive summary over the whole narrative including different parts of it, which seems accurate to me. Overall the tree seems balanced and it displays a much deeper structure with more levels of abstraction that can be utilized in the retrieval process. Taking a qualitative look at the responses generated by

[7] `https://github.com/parthsarthi03/raptor`

the new versus the old tree approach, I've observed that the new one identifies more nuances and provides a more comprehensive response than the original. For instance, when asked, "How did Cinderella achieve her happy ending?" the new approach generated an answer that displayed a deeper understanding of the prince's role in the story, whereas the old approach, while correct, seemed somewhat superficial. This observation suggests that the new tree structure offers a more detailed and insightful analysis, enhancing the depth of response interpretation.

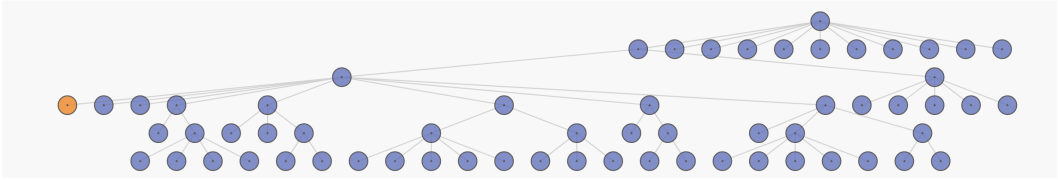

Figure 2: New RAPTOR tree for Cinderella story.

7 Conclusion

In the course of analyzing the current RAPTOR framework, I've compiled a series of improvement suggestions, as outlined in table 1. The primary limitations of the existing RAPTOR approach include the reliance on GMMs as the foundational clustering algorithm. This choice necessitates dimensionality reduction and typically results in the generation of flat tree structures. Furthermore, there is a notable absence of soft clustering for standard document sizes and a suboptimal implementation of sentence tokenization. Through my investigation, I found that agglomerative clustering, with its inherent tree-like hierarchical structure, serves as an ideal basis for constructing the envisioned RAPTOR retrieval tree. This method significantly simplifies the overall tree-building process, rendering the clustering component computationally negligible for document sizes typically encountered in academic papers. Moreover, positional embeddings have demonstrated considerable utility in capturing and conveying the structural nuances of documents, thereby enhancing the clustering algorithm's effectiveness. As a result of my work, there has been a comprehensive redesign of the tree-building and clustering processes, leading to a significant simplification and computational efficiency, particularly well-suited for textual applications. Qualitatively, this redesign facilitated the creation of deeper, more balanced retrieval tree structures and fostered a more profound understanding in the system's responses. Quantitatively, it consistently surpassed the original approach in handling both abstractive and extractive questions. However, the limitations of my project predominantly arise from the quantitative experiments using the QASPER dataset. In these experiments, the new approach exhibited performance declines in addressing yes/no questions and in determining "unanswerable" queries. Moreover, the experiments demonstrated considerable variability, which poses challenges to the reliability of the experiments precisely measuring possible performance enhancements. Generally, the QASPER F1 metric with its strict token-based comparisons, does not seem to effectively gauge the improved abstractive understanding of underlying documents in a retrieval system. Finally, given the time constraints, my evaluation was limited to using only the QASPER dataset for NLP papers, which restricts the generalizability of the findings across diverse datasets, contexts and different metrics.

As avenues for future work, it would enhance the validity of my findings to conduct additional experiments on varied datasets, such as those from medical, narrative, or legal fields, and to explore smarter evaluation methodologies beyond token-based metrics, for example, employing LLMs as model-based judges. Moreover, identifying and eliminating the causes behind the observed deviations in the experiments is crucial for further validation of my results. Further experimentation with applied hyperparameters in tree construction and positional embeddings could also be beneficial. Lastly, there are numerous potential modifications and enhancements that could be explored to improve the system. These include implementing proposed rotational position embeddings, exploring neural clustering approaches, and investigating ways to realize soft clustering to create a retrieval DAG instead of a tree. Additionally, enhancements on the retrieval side should be considered, including reevaluating tree retrieval methods and learned retrieval methods. Implementing weights, potentially through a neural approach, could further optimize retrieval performance and accuracy.

8 Ethics Statement

Looking at the original RAPTOR paper I did not find any ethical considerations and the reviewers for the ICLR conference did not raise a flag for a necessary ethics review of the paper. However, when looking at the topic with a broader perspective, I could think of a few societal risks that might arise from improved retrieval systems and RAG architectures in general. And I think it all comes down to who uses this advanced technology and for what purposes it is used. Let us say somebody builds a retrieval system with information about malicious activities, such as collections with detailed instructions for committing cyber crimes. Or even just using collections of information on cyber security to build a RAG retrieval system, but then using it for question and answering to get instructions to exploit weaknesses of IT systems. This exploitation by malicious actors to automate and maybe even scale harmful activities could provide a serious societal risk. Another consequence of advanced RAG systems for society could be the cessation of workplaces. The better RAG systems and LLMs generally become, the less knowledge workers might be needed in economy. The increased efficiency of these systems could raise fears in society about losing jobs and AI as being harmful or dangerous in general.

To address the issue of malicious intentions when using advanced RAG systems the connected LLMs could have built-in detectors for ill-intended prompts and refuse to provide requested answers. But this would only work in cases where malicious users do not control both the RAG system and the LLM they use. To address the issue of general societal fears about advancements in AI one could increase efforts to educate about real risks, opportunities and current technological possibilities, while trying to reduce the spread of misinformation and unreasonable fears.

References

Ekin Akyurek, Tolga Bolukbasi, Frederick Liu, Binbin Xiong, Ian Tenney, Jacob Andreas, and Kelvin Guu. 2022. Towards tracing knowledge in language models back to the training data. In *Findings of the Association for Computational Linguistics: EMNLP*. Available online at: `https://arxiv.org/abs/2205.11482`.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Leland McInnes, John Healy, and James Melville. 2020. Umap: Uniform manifold approximation and projection for dimension reduction.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations (ICLR)*. Available online at: `https://arxiv.org/abs/2401.18059`.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the Domain Adaptation of Retrieval Augmented Generation (RAG) Models for Open Domain Question Answering. *Transactions of the Association for Computational Linguistics*, 11:1–17.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding.

A Cluster Visualization

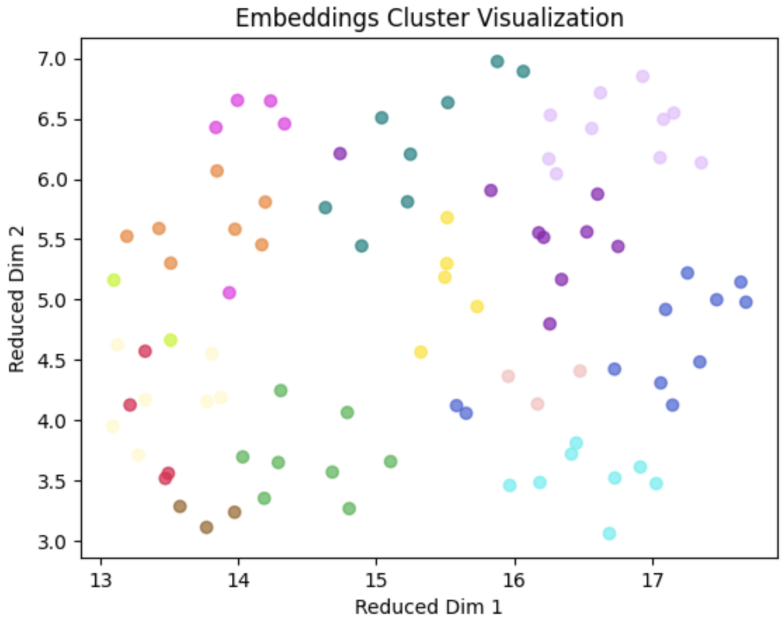

Figure 3: RAPTOR clustering with GMMs on initial text chunks from a NLP paper.

This visualization was created following the example code provided by OpenAI for embedding and cluster visualization[8]. As a preparatory step, I extracted the generated RAPTOR tree from an NLP paper in the QASPER dataset and retrieved the assignments of leaf nodes to their parents. With this mapping, I was able to modify the OpenAI script to visualize the clustering of RAPTOR for the leaf nodes. Additionally, I adapted the dimensionality reduction to use Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) from McInnes et al. (2020) to replicate the approach used in the original RAPTOR paper, hoping that the 2D representation would provide greater insight into what actually occurs during RAPTOR's clustering. However, it should be noted that the 2D representation, resulting from dimensionality reduction, is merely an abstraction of the embeddings and clustering, and not an exact depiction of the underlying events.

[8] `https://cookbook.openai.com/examples/clustering`

B Agglomerative Clustering Dendrogram

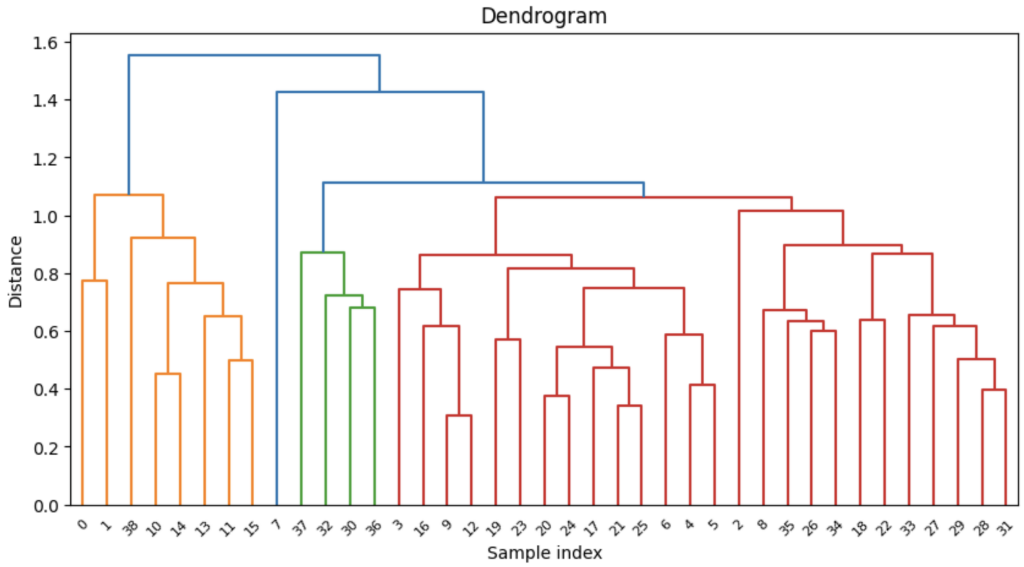

Figure 4: Dendrogram of Cinderella story using cosine distances and average linking.