Strategies for Efficient Retrieval-augmented Generation in Clinical Domains with RAPTOR: A Benchmarking Study

Xumou Zhang[12], **Qixuan Hu**[12], **Jinman Kim**[1], **Adam G. Dunn**[2]
[1]Faculty of Engineering, University of Sydney
[2]Faculty of Medicine and Health, University of Sydney
{xumou.zhang,qixuan.hu,jinman.kim,adam.dunn}@sydney.edu.au

Abstract

The Recursive Abstractive Processing for Tree-Organized Retrieval (RAPTOR) framework deploys a hierarchical tree-structured datastore to integrate local and global context, enabling efficient handling of long documents for language models. This design is especially useful when cloud-based language models are unavailable or undesirable. For instance, with offline confidential patient records or stringent data-privacy requirements. We benchmarked RAPTOR on the QuALITY dataset and a novel Clinical Trial question-answering dataset (CTQA) drawn from over 500 000 registry entries. Experiments varied question complexity (simple vs. complex), four language models, four embedding models, and three chunking strategies. Also incorporated GPT-4o as a cloud-based baseline. Results show that, with optimal settings, RAPTOR combined with smaller local models outperforms GPT-4o on complex CTQA questions, although this gain does not extend to QuALITY. These outcomes highlight RAPTOR's promise as a practical, locally implementable solution for long-context understanding.

1 Introduction

Health applications of language models cover a wide range of clinical tasks, but a recent systematic review (Bedi et al., 2024) found that only a small proportion use patient data, and evaluation with real users is limited. Some of the challenges of broader use include data privacy (Ullah et al., 2024; Das et al., 2024), the need to produce stable and repeatable results (Meskó and Topol, 2023), and the cost efficiency of using cloud-based language models as a service (Chen et al., 2023a).

The local language models could be a suitable direction for health domain users in this scenario. It helps them to implement applications and store data on a local machine. However, this also means that language models often need to be implemented on consumer hardware, limiting the size of the models that can be used. This sentiment is echoed by Mesko and Topol (Meskó and Topol, 2023), who explain that locally hosted models enhance both the privacy and robustness of clinical language models. Language models such as Mistral-7B-Instruct-v0.2 (Jiang and et al., 2023) can be deployed on consumer hardware but are affected by the long context problem. The long context models like Llama-3.1-8B-Instruct (AI@Meta, 2024) can still suffer from context length when dealing with long bioinformatical or clinical reports.

Several approaches have been developed to address the issues of handling longer documents in text summarisation and generation. LongLoRA (Chen et al., 2024b) and Infini-attention (Munkhdalai et al., 2024) modify the attention mechanism and fine-tune the models to enhance the model capability of handling long texts up to 500K tokens in length. However, both of them require additional fine-tuning in the training phase and extra memory during inference. Others including CFIC (Qian et al., 2024a) and BGE Landmark Embedding (Luo et al., 2024), introduce special layers or embedding models to select partial information without breaking down the long contents. These two methods also require additional training to run properly, and are limited to either 32k token max length or the max model context length, respectively.

Retrieval-augmented Generation (RAG) framework (Lewis et al., 2020), which works by setting up an external datastore to keep the long contents in chunks, and retrieving partial information from the preprocessed datastore as a reference text when generating a response. It is a cost-effective method and able to deal with long content, but potential information loss is an issue during the text retrieval process.

Recursive Abstractive Processing for Tree-Organized Retrieval (RAPTOR) (Sarthi et al.,

Proceedings of Recent Advances in Natural Language Processing,pages 1420–1429
Varna, Sep 8–10, 2025
https://doi.org/10.26615/978-954-452-098-4-164

2024) extends from RAG and is an efficient method for language models to handle long text contents and reduce model hallucination without any internal model modifications. Different from the naive RAG framework (Lewis et al., 2020), it consists of a tree-structured datastore to preprocess and retrieve both relevant local and overview information from it to support model generation. It can be more reliable when the user query requires information from a distance. RAPTOR has been shown to outperform other frameworks that do not use a tree structure for QuALITY dataset (Pang et al., 2022), which makes it an appropriate framework to evaluate for long context information sources.

Different from other tree-based RAG-based frameworks like T-RAG (Fatehkia et al., 2024) and RAGAR (Khaliq et al., 2024), the RAPTOR framework implements the tree structure directly, and the retrieved text can contain complete overview information as support. In contrast, T-RAG implements the tree structure to modify the user query for organisational-related entities, and RAGAR uses the tree structure to generate sub-questions for the original user query. None of them are capable of dealing the overview information loss like RAPTOR does.

RAPTOR has shown to achieve state-of-the-art performance on the QuALITY dataset (Pang et al., 2022), which is a question-answering dataset aiming for long-context scenarios. While RAPTOR has achieved strong results, it was evaluated on the cloud-based GPT-4 model. To our knowledge, its ability to work with smaller language models like Mistral-7B-Instruct-v0.2 has not been validated.

Our aim was to benchmark the RAPTOR framework using language models that can be implemented locally. The contributions of this study are as follows:

- Benchmarking the response accuracy of the RAPTOR framework using both cloud-based and efficient locally implemented language models.
- Showing that the RAPTOR approach can be configured to outperform GPT-4o specifically for complex questions, where answers synthesise distant information.
- Evaluation of two efficient semantic chunking strategies for the RAPTOR framework, which make use of semantic relationships between texts.

2 Background and related work

2.1 RAPTOR framework

The workflow of the RAPTOR framework comprises the following steps to produce a tree structure: (1) split the document text into a set of chunks; (2) use an embedding model to create sentence embeddings for each chunk; (3) use the Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) (Leland et al., 2018) to reduce the dimensionality; (4) use the Gaussian Mixture Model (Reynolds et al., 2009) with Bayesian information criterion to cluster current chunks; (5) concatenate the texts within each cluster and generate a detailed summary using a language model; (6) repeat the above process until the final root node is reached.

The model is then able to either follow the tree structure to retrieve relevant information at any step or collapse the tree structure to gather relevant information directly. This means the RAPTOR framework can potentially use both overview and local information as a reference during the inference process when generating a response to a query.

2.2 Chunking strategies in RAPTOR

In the RAPTOR framework, methods for splitting the document into chunks can affect the performance of the model. Chunking methods include character chunking and semantic chunking.

2.2.1 Chunking by character

The naive RAG chunking splits text using special characters in the text (e.g. “.” or “;”) under the preset maximum length limit. This was used in the RAPTOR framework. It is efficient in execution time and computational cost. However, since it does not consider semantic meanings, it can potentially over-split, which may result in information loss or inconsistent chunks.

2.2.2 Chunking by semantic information

One approach for improving chunking is to make use of semantic differences over the length of the document. This unsupervised approach converts sentences into embedding vectors, calculates the cosine distance between adjacent sentences, and then splits the document into chunks using a threshold value for the cosine distance. This approach can be used directly within the RAPTOR framework in the first step.

Related researches focus on adapting deep learning models to support text segmentation. One proposes a BERT-based embedding model to convert sentences into sentence embeddings, then using those as inputs to a transformer model to identify breakpoints in the sentence embedding (Lukasik et al., 2020). Another study proposes the use of a moving window inside the BERT model structure (Zhang et al., 2021). A PoNet-based model (Tan et al., 2021) was able to extend the context length from 512 tokens to 4096 tokens. Note that while these approaches were not designed specifically for use within the RAG framework, the goal is the same, and the methods can be adapted for use with RAPTOR.

2.2.3 Chunking by language model

A recent set of approaches considers the use of language models to support chunking. One example proposes splitting a document into chunks called 'propositions' and using prompt engineering pipelines to group propositions together using a node-tree structure (Chen et al., 2023b). Another example proposes the use of language models as decision-making machines that select different text chunks for RAG-based tasks (Qian et al., 2024b). While these approaches represent elegant solutions for chunking, the current limitation is the computational and time costs, which may make them less practical for downstream tasks and especially for application domains where consumer-level hardware is a constraint.

2.3 Embedding models in RAPTOR

Embedding models can have a major impact on the performance of downstream tasks. RAPTOR uses embedding models in both datastore construction and the information retrieval process. A recent study compared the performance of several embedding models for downstream tasks in the health domain, including general embedding models and specialised models trained using text data from health application domains (Excoffier et al., 2024). The results show large variations in performance across the embedding models.

The context lengths of models can also affect performance on downstream tasks, even when used within an RAG framework. For example, BioBERT (Lee et al., 2020) is a popular model trained on the biomedical text and has a maximum context length of 512. This compares to the more recently developed models jina-embeddings-v2-base-en (Jina) (Günther et al., 2023), BGE-M3 (BGE) (Chen et al., 2024a), and text-embedding-ada-002 (Neelakantan and et al., 2022), which have a maximum context length of 8,192. The way this can affect performance is when shorter context length embeddings cut off sentences that exceed the maximum context length, leading to information loss.

2.4 Language models in RAPTOR

Language models with between 1 billion and 10 billion parameters are considered to be 'medium scale' (Minaee et al., 2024). In 2024, the most powerful consumer-level GPUs available on the market have 24GB of GPU memory. This limits the scale of language models that can be implemented comfortably to these models with 10 billion parameters or fewer. The Mistral-7B-Instruct-v0.2 model and Llama-3.1-8B-Instruct model each have approximately 7 billion and 8 billion parameters, respectively. Compared to large-scale models like GPT-4 and Gemini Pro, medium-scale models are more practical for use in application domains where local computing resources are required.

3 Methodology

The RAPTOR framework's three modules - the chunking strategy, the embedding model and the summarization/generation model are presented in Table 2, then implemented and evaluated for both the CTQA and the QuALITY datasets as presented in Figure 1. For reference, we also measured GPT-4o model's zero-shot accuracy on each dataset with same prompting.

3.1 Datasets

Benchmarking was performed using two datasets. The QuALITY dataset (Pang et al., 2022) is commonly used to evaluate approaches in the RAG framework. We introduce the Clinical Trials Question and Answer (CTQA) dataset as an example of a large dataset from biomedical application domains. Both datasets include opportunities to ask simple and complex questions (see examples below). Note that QuALITY has multiple-choice questions and answers and CTQA has short-answer questions and answers.

3.1.1 QuALITY dataset

QuALITY dataset (Pang et al., 2022) was used in the original evaluation of the RAPTOR framework (Sarthi et al., 2024). The dataset includes

Type	Content
QuALITY, Simple	Which of the following most closely fits the theme of this article?
QuALITY, Complex	What does the author likely think will happen if democracy does not evolve?
CTQA, Simple	In the results of this trial, how many participants had serious adverse events in each study arm?
CTQA, Complex	In the design of this study, what intervention was used in the control arm?

Table 1: Examples of simple & complex questions in the QuALITY and CTQA datasets

three main sections: the main text documents, questions and correct answers. It was used as three subsets: training, development, and testing. The development subset included 230 articles and multiple-choice questions, where each question was labelled as simple or complex 1. Main text documents contain from 2000 tokens to 6000 tokens. In the original Quality dataset study, the author states that the difficulty of the questions is labelled manually, where the question is labelled as a hard/complex question if the human annotator cannot answer the question within 45 seconds.

3.1.2 CTQA dataset

ClinicalTrials.gov is a registry for more than 500,000 trials and other studies, designed to provide public information about the design of studies before the study begins (cli, 2024b,a). Each study includes sections of text with a summary of the study, the population, interventions or exposures, and outcome measures. Some studies on ClinicalTrials.gov also include tables with numerical data describing the summary results of the study after it is completed. With a similar structure to the QuALITY dataset, CTQA includes main text documents (the registry entry), and pairs of short answer questions with their answers. We extracted the trial registrations and converted them into the CTQA dataset. It is a large dataset and grows over time, has a complex structure including structured and unstructured data, and is an application domain representing a very large and expensive industry domain. Downstream tasks include those related to improving the efficiency of trial designs to avoid redundancy and avoid termination, synthesis and meta-analysis of trials that answer the same clinical question, and checking for reporting bias when results in published trial articles do not match what was registered.

The simple question for the CTQA dataset was focused on information extraction where the correct answer can be extracted at scale from the results section if the study have one; the complex question requires both information extraction and generating an appropriate response using contextual information, which requires more steps for answer finding (Table 1). This is similar to an approach used in a previous study (Jeong et al., 2024).

3.2 Chunking strategy configuration setup

The three chunking strategies included a character-based chunking strategy, a new semantic chunking strategy, and a chunking strategy that uses a deep learning method.

3.2.1 Naive character chunking

The default chunking strategy used in the RAPTOR framework is the naive character chunking strategy (character chunking), as the RAPTOR framework was not focused on this aspect. In this study, the default character-splitting chunking strategy will also be tested on both CTQA and QuALITY datasets as the reference.

3.2.2 Recursive moving percentile semantic chunking

We introduce a modified semantic chunking strategy named Recursive Moving Percentile Semantic Chunking (RMP chunking). Rather than using a globally fixed value for the breakpoint between text sections, we calculate the threshold value dynamically with the moving percentile. The process is divided into the following steps: (1) split the original document into the minimum chunks using character chunking; (2) convert chunks into embeddings via a preset embedding model and calculate the distances for adjacent pieces with the context padding; (3) calculate moving percentile values based on the distances with the preset window size and the preset percentile threshold value, and label all critical indices that exceed the threshold value; (4) repeat from step one until there are no more indices within chunks that exceed the threshold value.

3.2.3 Deep learning based semantic chunking

PoNet (Tan et al., 2021) is a chunking strategy that uses a deep learning model and comes from the

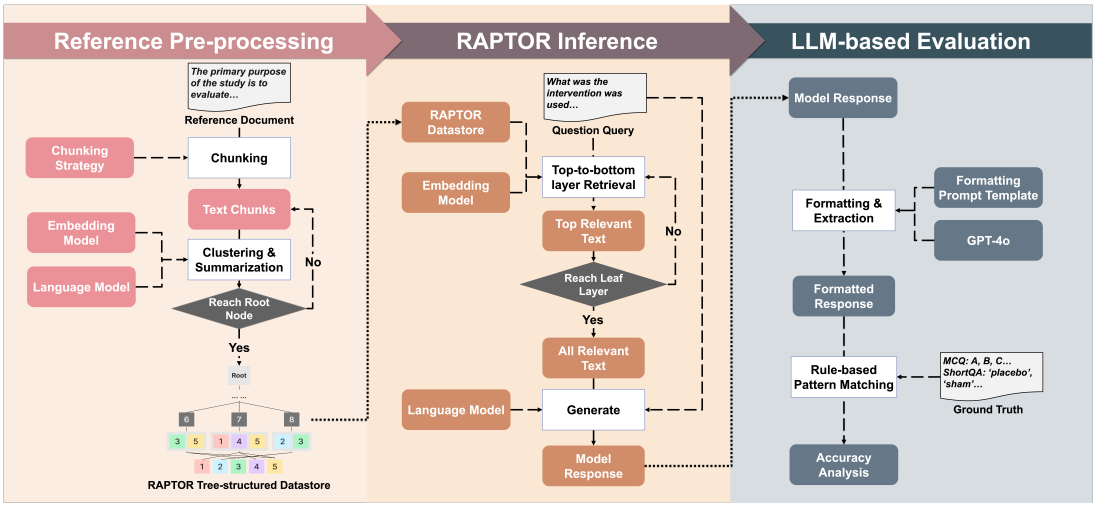

Figure 1: Overall structure diagram of the framework and evaluation process. The workflow contains 3 parts: (1) reference pre-processing, which constructs the tree-structured datastore using the reference document; (2) RAPTOR inference, which contains the information retrieval and the model generation process; (3) LLM-based evaluation, which demonstrates how the raw model responses are cleaned and scored.

research area of text segmentation. While text segmentation models like hierarchical BERT (Zhang et al., 2021) and PoNet were not designed to fit the needs of the chunking strategies in RAG-based frameworks, their motivations and structures are similar and can be used as chunking strategies in RAG-based frameworks.

3.3 Embedding model configuration setup

Four embedding models were used in the experiments 2. The baseline model is the text-embedding-ada-002 (OpenAI Embedding) model from OpenAI with 8192 context length which deployed in the cloud-based server. 2 local embedding models with long context length are introduced into the experiment to deal with longer text: the BGE-M3 (BGE) Embedding model and the jina-embeddings-v2-base-en (Jina) Embedding model; the BGE Embedding model is utilized for the RAG implementation according to their technical report, and Jina Embedding model has also been utilized to extend the context length from 512 to 8192 tokens. Also, a biomedical domain embedding model BioBERT with a 512 context length is also introduced to the experiment for comparison.

3.4 Language model configuration setup

Four language models were configured in the benchmarking experiments as well. The GPT-3.5-Turbo model is here to represent the cloud-based very large language models. The Mistral-7B-Instruct-v0.2 model and Llama-3.1-8B-Instruct model represent the local medium language models before and after the era of long-context models, respectively. The Llama-3.2-1B-Instruct is representing the local small-size language model for the complete comparison.

To simulate a realistic low-setting environment, the experiments all used the 4-bit QLoRA quantization setting for the local models.

3.5 LLM-based evaluation with the language model and the rule-based method

The model responses do not always match the correct format in terms of verbosity, even though they may still be correct (e.g., a response of "neither study arm had any serious adverse events" vs. the expected "Arm 1: 0; Arm 2: 0"), the pattern matching could be inaccurate, and the manual process would be time-consuming.

We designed an LLM-as-Judge approach using LLM for format extraction and rule-based pattern matching for accuracy scoring to overcome those issues. While our evaluation is consistent to others (Hashemi et al., 2024), in that we use LLM to judge the results using questionnaires and collect scores in rule-based evaluation, our method is able to work for both MCQs and short QAs since we focus on if the response matches the answer. We confirmed that the LLM-based approach was accurate with a manual assessment of 500 examples of MCQ from the QuALITY dataset and 500 short QA examples from the CTQA dataset, achieving a match rate of 98.8%. The process is as follows: (1)

Component	Name
Chunking Strategy	Naive Character Chunking Recursive Moving Percentile Semantic Chunking PoNet Semantic Chunking
Embedding Model	text-embedding-ada-002 BGE-M3 jina-embeddings-v2-base-en BioBERT
Language Model	GPT-3.5-Turbo Mistral-7B-Instruct-v0.2 Llama-3.1-8B-Instruct Llama-3.2-1B-Instruct

Table 2: Component Configurations

generate model responses using the prompt templates provided in ZeroSCROLLS (Shaham et al., 2023) and the framework in Figure 1; (2) extract the formatted answer from the raw model response using the GPT-4o model with a 3-shot prompt template; (3) evaluate the extracted answer with the ground truth with rule-based pattern matching.

4 Results

4.1 Experimental results for the CTQA dataset

The varied across chunking strategies, embedding models, language models, and across simple and complex questions for the CTQA dataset (Table 3 and Figure 2).

For simple questions, PoNet chunking was best on GPT-3.5-Turbo and Llama-3.2-1B-Instruct, beating the other strategies in 75% of trials by 8.46% and 5.84% on average; while RMP chunking led on Mistral-7B-Instruct and Llama-3.1-8B-Instruct by 17.42% and 10.22%. For complex questions, PoNet chunking topped all four models, with average accuracy gains of 1.42%, 9.89%, 6.87% and 6.63%.

Among the 4 embedding models, OpenAI and BGE models achieved the highest accuracy. Compared to other two embedding models, the result shows that OpenAI and BGE models lead ahead in the simple question by 52.91% and 75.71% on average respectively; and lead ahead in complex question by 17.19% and 9.66% on average respectively.

In the direct comparison between the four language models and with the separate GPT-4o model, the simple question results show that the GPT-4o substantially outperformed the RAPTOR framework configurations. GPT-4o reached 93.25% accuracy, compared to an average of 52.67% for the GPT-3.5-Turbo model, 43.49% for the Mistral-7B-Instruct-v0.2 model, 46.26% for the Llama-3.1-8B-Instruct model and 31.89% for the Llama-3.2-1B-Instruct model across the different configurations. The result was different for the complex question, where the GPT-4o mode achieved 52.26% accuracy, compared to an average of 70.01% for the GPT-3.5-Turbo model, 59.97% for the Mistral-7B-Instruct-v0.2 model, 74.98% for the Llama-3.1-8B-Instruct model and 57.77% for the Llama-3.2-1B-Instruct model across the set of tested configurations within the RAPTOR framework.

4.2 Experimental results for the QuALITY dataset

In the QuALITY dataset experiments, the performance difference between configurations are relatively small.

In terms of language models, GPT-3.5-Turbo and Mistral-7B-Instruct-v0.2 model lead ahead in both simple and complex questions, the Llama-3.1-8B-Instruct model falls behind in the second tier, and leaves Llama-3.2-1B-Instruct model at last. The result shows 67.24%, 66.87%, 60.30% and 34.32% on average for the 4 models respectively for the simple questions; 49.36%, 48.86%, 43.72% and 28.93% on average respectively for the complex questions.

Performance on different chunking strategies showed consistent trends. RMP chunking generally outperformed others in the simple and complex questions in most of language models. Average accuracy for the simple question using the RMP chunking strategy was 68.27%, 68.12%, 63.88% and 34.33% for 4 language models respectively. Average accuracy for the complex question using the RMP chunking strategy was 50.97%, 51.24%, 46.31% and 28.94% for 4 language models respectively.

The OpenAI and BGE embedding models outperformed Jina and BioBERT embedding models across all but one of the configurations (Table 3). Overall differences between OpenAI and BGE are relatively small, suggesting that the choice of embedding model between the two is less important than the choice of language model and chunking strategy.

The GPT-4o model outperformed RAPTOR

Model Configuration		Simple Question (n=400/1021)			Complex Question (n=400/1065)		
Language Model	**Embedding Model**	**Char. Chunking**	**RMP Chunking**	**PoNet Chunking**	**Char. Chunking**	**RMP Chunking**	**PoNet Chunking**
GPT-3.5-Turbo	OpenAI	63.44/69.44	**64.37/70.03**	54.05/69.79	75.87/49.86	73.17/51.27	**77.12/51.97**
	BGE	56.37/67.68	64.25/**68.27**	**78.48**/67.87	70.38/50.42	77.70/**52.96**	**74.52**/50.05
	Jina	36.12/**68.95**	50.42/68.66	**54.61**/68.17	48.78/50.23	69.32/**50.33**	**72.55**/47.98
	BioBERT	37.00/61.51	21.59/**66.11**	**51.34**/60.33	63.41/44.79	67.82/**49.30**	**69.48**/43.19
Mistral-7B-Inst.	OpenAI	55.05/68.66	**64.21**/69.64	29.37/**69.70**	68.42/48.64	**74.23/52.58**	74.03/47.17
	BGE	39.87/67.48	58.79/67.68	**72.27/66.70**	52.04/50.33	54.21/**52.77**	**69.53**/49.30
	Jina	39.52/66.90	**50.75/69.93**	35.81/68.95	45.62/49.39	52.11/**51.36**	**59.35**/48.83
	BioBERT	6.00/60.72	15.79/**65.23**	**23.96**/60.82	53.41/43.47	**60.53/48.26**	56.12/44.23
Llama-3.1-8B-Inst.	OpenAI	55.79/60.14	**65.74/65.81**	35.87/60.50	77.03/43.76	84.72/**48.26**	**82.60**/45.44
	BGE	48.44/62.29	69.93/**63.07**	**79.76**/61.40	70.03/43.85	79.70/**45.54**	**80.50**/42.72
	Jina	45.02/58.37	**61.11/65.72**	44.74/60.04	68.64/42.07	69.47/**46.38**	**79.22**/42.72
	BioBERT	6.78/53.57	12.5/**60.92**	**29.5**/51.81	62.37/40.00	66.67/**45.07**	**78.88**/38.87
Llama-3.2-1B-Inst.	OpenAI	34.44/33.14	**40.89/34.57**	33.03/34.24	55.94/29.73	58.99/29.20	**63.46/29.91**
	BGE	33.27/35.75	38.11/**34.77**	**45.63**/33.20	51.57/27.79	62.96/**28.45**	**66.87**/28.35
	Jina	23.25/**37.22**	30.46/33.79	**32.73**/35.16	54.70/**29.29**	61.67/28.64	**63.22**/29.20
	BioBERT	19.73/**34.47**	22.70/34.18	**28.47**/31.34	43.90/28.17	52.13/**29.48**	**57.86**/28.92
GPT-4o	–	–	–	93.25/95.00	–	–	52.26/83.29

Table 3: Task accuracy of the RAPTOR framework with the CTQA/QuALITY datasets. n = the number of questions in CTQA/QuALITY; **Bold** = highest among 3 chunkings; Underline = highest among 4 embeddings.

in the simple and complex questions by around 30% (Table 3 & Figure 2).

5 Discussion

The results show that differences in the chunking strategy, embedding model, and the complexity or type of questions each appear to have an impact on performance within a RAPTOR framework. These choices will likely be especially important in scenarios where implementation is restricted to local machines and access to high-performance cloud computing is restricted. The performance difference of GPT-4o in simple and complex questions in both datasets suggests that our approach to distinguish question complexity is correct. Also, the lower performance of GPT-4o compared to a smaller language model using the RAPTOR framework suggests that for complex questions where contextual and local information are important, it is not enough to only increase the size of the language model.

5.1 Chunking strategy

The results show that for a given model, RMP and PoNet chunking strategies generally improve the performance relative to standard character chunking in the CTQA dataset. These differences appeared for both simple and complex questions. The chunking strategy result may seem counterintuitive because the RAPTOR framework is designed to 'connect' all the information together in a tree structure. However, the main benefit of chunking strategies in the RAPTOR framework may come from improved summarisation in the leaf nodes, and this improvement then flows through the local and global information in the tree structure.

Lower performance was found for the PoNet chunking strategy using the OpenAI embedding model with all four models in the simple question of the CTQA dataset. Because the PoNet chunking does not have a hard upper limit in the maximum chunk size, it could have larger initial chunks. This suggests that the OpenAI embedding model may not perform as well as the BGE embedding model in the long context embedding task.

5.2 Language model

Although there are performance differences between the GPT-3.5-Turbo model and the other three models, the results show that both the Mistral-7B-Instruct-v0.2 model and Llama-3.1-8B-Instruct model are able to reach or surpass the same level of performance as the GPT-3.5-Turbo with the BGE embedding model and the semantic chunking strategies. It shows that the RAPTOR framework is able to narrow the performance gaps between large-scale models and smaller models, and demonstrates that the RAPTOR framework can substantially mitigate the performance gaps due to the context length differences between the models.

GPT-4o seems to fall behind in complex questions in the CTQA dataset. Note that when experts answer the complex CTQA question, they typically make use of the information from the title, brief and detailed summary, and contextual information about the structure and design of the trial. This suggests that the ability to synthesise local and global information is particularly important for the question, and may explain performance differences

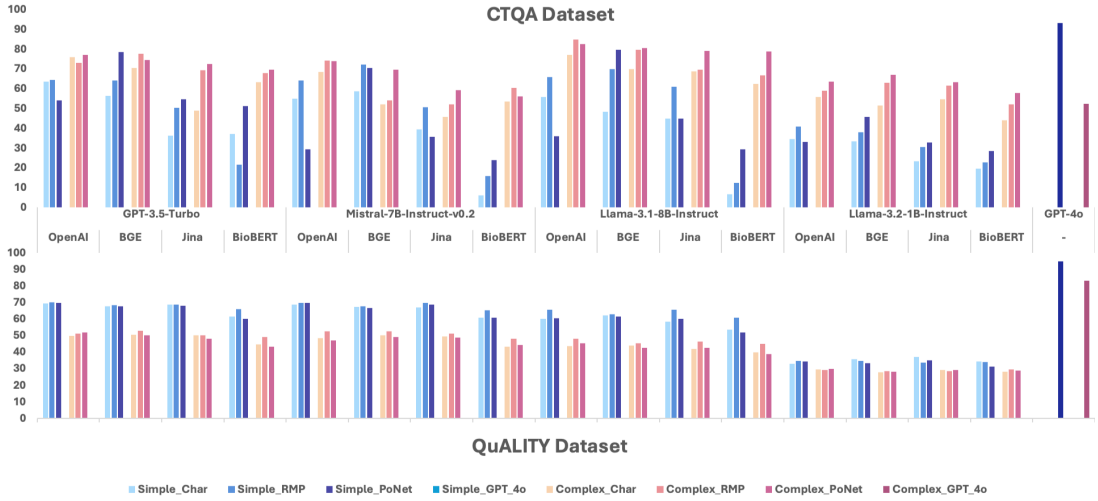

Figure 2: Task accuracy for the CTQA (top) and QuALITY (bottom) dataset across simple and complex questions with different configurations, showing differences in performance compared to a GPT-4o baseline.

between GPT-4o and the top-performing RAPTOR configuration.

5.3 Embedding model

The results show that the OpenAI embedding model and the BGE embedding model achieved similar performance, with few exceptions. The BGE model was designed to handle long contextual tasks natively, which may explain the generally strong performance. When choosing an embedding model for use with the RAPTOR framework, the BGE embedding model appears to perform at least as well as the OpenAI embedding model and it is feasible to fine-tune the BGE model for domain adaptations.

5.4 Future work

The results suggest that applications of the RAPTOR framework and appropriate configurations enable the small-size language model to perform comparably to the larger models, and may still be relevant even as larger language models are developed. A further opportunity could be to apply the RAPTOR framework as a file management system locally using smaller language models.

Another interesting aspect of this work was the use of simple and complex questions. An underlying assumption is that the RAPTOR framework may be better suited to questions that need to make use of local and global information. 'Complex' questions in the QuALITY dataset do not always represent this complexity. A future opportunity might be to consider multiple approaches that work to evaluate the complexity of the question and then auto-configure the best configuration/structure that will yield the best answer.

ClinicalTrials.gov is underutilised. It is a large public dataset with a broad range of downstream tasks (Long et al., 2023; Wang et al., 2022; Elkin and Zhu, 2021). Many of these may benefit from RAG-based methods to support information extraction, synthesis, and classification.

6 Conclusion

This benchmarking study showed that the RAPTOR framework varies in performance depending on configuration and the types of questions it is used to answer. The results showed that language models with fewer than 10 billion parameters can be used with the RAPTOR framework to overcome the long context problem, and these configurations are a feasible solution in scenarios where larger language models cannot be used. Where the RAPTOR framework can be implemented on consumer-level hardware, it provides more privacy and configuration control for users, which is likely to be important for a range of medical-related tasks. The results also showed that the use of a semantic chunking method improved the results compared to the standard character chunking method.

7 Acknowledgements

Funding for this research is from National Institutes of Health R01LM012976.

References

2024a. Protocol registration data element definitions for interventional and observational studies. Last updated on June 17, 2024.

2024b. Study data structure. Last updated on April 01, 2024.

AI@Meta. 2024. Llama 3 model card.

Suhana Bedi, Yutong Liu, Lucy Orr-Ewing, Dev Dash, Sanmi Koyejo, Alison Callahan, Jason A Fries, Michael Wornow, Akshay Swaminathan, Lisa Soleymani Lehmann, et al. 2024. Testing and evaluation of health care applications of large language models: A systematic review. *JAMA*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023a. Frugalgpt: How to use large language models while reducing cost and improving performance.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Dong Yu, and Hongming Zhang. 2023b. Dense x retrieval: What retrieval granularity should we use? *arXiv preprint arXiv:2312.06648*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024b. Longlora: Efficient fine-tuning of long-context large language models.

Badhan Chandra Das, M Hadi Amini, and Yanzhao Wu. 2024. Security and privacy challenges of large language models: A survey. *arXiv preprint arXiv:2402.00888*.

Magdalyn E Elkin and Xingquan Zhu. 2021. Predictive modeling of clinical trial terminations using feature engineering and embedding learning. *Scientific reports*, 11(1):3446.

Jean-Baptiste Excoffier, Tom Roehr, Alexei Figueroa, Michalis Papaaioannou, Keno Bressem, and Matthieu Ortala. 2024. Generalist embedding models are better at short-context clinical semantic search than specialized embedding models. *arXiv preprint arXiv:2401.01943*.

Masoomali Fatehkia, Ji Kim Lucas, and Sanjay Chawla. 2024. T-rag: lessons from the llm trenches. *arXiv preprint arXiv:2402.07483*.

Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, Maximilian Werk, Nan Wang, and Han Xiao. 2023. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents.

Helia Hashemi, Jason Eisner, Corby Rosset, Benjamin Van Durme, and Chris Kedzie. 2024. Llm-rubric: A multidimensional, calibrated approach to automated evaluation of natural language texts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13806–13834.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity.

Albert Q. Jiang and et al. 2023. Mistral 7b.

M Abdul Khaliq, P Chang, M Ma, Bernhard Pflugfelder, and F Miletić. 2024. Ragar, your falsehood radar: Rag-augmented reasoning for political fact-checking using multimodal large language models. *arXiv preprint arXiv:2404.12065*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

McInnes Leland, Healy John, and Melville James. 2018. Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

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

Bowen Long, Shao-Wen Lai, Jiawen Wu, and Srikar Bellur. 2023. Predicting phase 1 lymphoma clinical trial durations using machine learning: An in-depth analysis and broad application insights. *Clinics and Practice*, 14(1):69–88.

Michal Lukasik, Boris Dadachev, Gonçalo Simoes, and Kishore Papineni. 2020. Text segmentation by cross segment attention. *arXiv preprint arXiv:2004.14535*.

Kun Luo, Zheng Liu, Shitao Xiao, and Kang Liu. 2024. Bge landmark embedding: A chunking-free embedding method for retrieval augmented long-context large language models. *arXiv preprint arXiv:2402.11573*.

Bertalan Meskó and Eric J Topol. 2023. The imperative for regulatory oversight of large language models (or generative ai) in healthcare. *NPJ digital medicine*, 6(1):120.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. 2024. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*.

Arvind Neelakantan and et al. 2022. Text and code embeddings by contrastive pre-training.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel Bowman. 2022. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358, Seattle, United States. Association for Computational Linguistics.

Hongjin Qian, Zheng Liu, Kelong Mao, Yujia Zhou, and Zhicheng Dou. 2024a. Grounding language model with chunking-free in-context retrieval. *arXiv preprint arXiv:2402.09760*.

Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Yujia Zhou, Xu Chen, and Zhicheng Dou. 2024b. Are long-llms a necessity for long-context tasks? *arXiv preprint arXiv:2405.15318*.

Douglas A Reynolds et al. 2009. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663).

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations (ICLR)*.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. Zeroscrolls: A zero-shot benchmark for long text understanding. *arXiv preprint arXiv:2305.14196*.

Chao-Hong Tan, Qian Chen, Wen Wang, Qinglin Zhang, Siqi Zheng, and Zhen-Hua Ling. 2021. Ponet: Pooling network for efficient token mixing in long sequences. *arXiv preprint arXiv:2110.02442*.

Ehsan Ullah, Anil Parwani, Mirza Mansoor Baig, and Rajendra Singh. 2024. Challenges and barriers of using large language models (llm) such as chatgpt for diagnostic medicine with a focus on digital pathology–a recent scoping review. *Diagnostic Pathology*, 19(1):1–9.

Siyang Wang, Simon Šuster, Timothy Baldwin, and Karin Verspoor. 2022. Predicting publication of clinical trials using structured and unstructured data: model development and validation study. *Journal of Medical Internet Research*, 24(12):e38859.

Qinglin Zhang, Qian Chen, Yali Li, Jiaqing Liu, and Wen Wang. 2021. Sequence model with self-adaptive sliding window for efficient spoken document segmentation. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 411–418. IEEE.