RAPTOR: RECURSIVE ABSTRACTIVE PROCESSING FOR TREE-ORGANIZED RETRIEVAL

Vishnu Vardhan Reddy Kandada (21CS002383)

Supervisor: Prof. Utsav Upadyay

INTRODUCTION:

With the huge amount of information available today, getting meaningful insights from large texts can be challenging. Current language models like BERT and GPT-3 work well with short texts but struggle with long documents due to their limited input size. RAPTOR (Recursive Abstractive Processing for Tree-Organized Retrieval) is a new method designed to handle large documents better. It uses a tree structure to organize and summarize text, making it easier to retrieve relevant information from long documents.

LITERATURE SURVEY:

Recent advancements in text summarization and document processing have focused on handling short and medium-length texts effectively. However, models like **BERT (Devlin et al., 2019)** and **GPT (Brown et al., 2020)** face significant challenges when dealing with large documents due to their quadratic complexity and inability to maintain context over long sequences. BERT's attention mechanism, for example, limits its capacity to process long documents in a single pass, which leads to fragmented contextual understanding. Similarly, GPT models, while impressive for generating text, struggle to retrieve information efficiently from extensive documents. Efforts to address these challenges, such as **Longformer (Beltagy et al., 2020)** and **BigBird (Zaheer et al., 2020)**, have introduced sparse attention mechanisms that reduce computational load, allowing models to process longer documents. However, these models still face difficulties in maintaining a coherent semantic understanding across large documents, particularly when processing multi-level structures such as chapters or sections.

On the summarization front, recursive approaches such as **Hierarchical Attention Networks (HAN)** (Yang et al., 2016) have been employed to break documents into smaller chunks for processing. These models attempt to summarize paragraphs or sections individually before aggregating them into a larger summary, but they often fail to capture the full semantic flow across the entire document. Similarly, extractive summarization models like **TextRank (Mihalcea & Tarau, 2004)** and **LexRank**

(Erkan & Radev, 2004) rely heavily on keyphrase extraction but lack the depth to summarize large texts effectively.

The **RAPTOR model** aims to fill these gaps by introducing a **hierarchical and recursive document processing framework**. This method allows for efficient summarization and querying of large documents by recursively summarizing sections while maintaining the semantic flow across the entire document. RAPTOR's multi-level approach to document summarization, combined with its ability to answer queries in real-time, makes it a significant improvement over traditional models like BERT and GPT for large-scale document analysis.

Unlike previous approaches, RAPTOR leverages hierarchical structure to retain context, which is particularly crucial when dealing with large documents, making it more adept at both summarization and query-based tasks. This project seeks to address the current limitations by providing a scalable solution for handling long documents in real-time, which is a notable advancement over existing models in the field.

RESEARCH GAP: -

Current language models and retrieval systems face several key challenges when dealing with large documents:

1. **Limited Input Size**: Models like BERT and GPT-3 have a fixed input size, typically limited to a few hundred tokens. This restriction means they can't process or understand lengthy documents in their entirety, often missing crucial context and information located in different sections of the text.

2. **Single-Level Understanding**: Traditional retrieval models like BM25 and even advanced neural retrieval systems do not support multi-level abstraction or recursive processing. They work on a single level of granularity, which limits their ability to understand and retrieve information that requires a deeper, layered understanding of the document.

3. **Inefficient Retrieval**: Existing summarization and retrieval models lack a structured approach to navigate and extract information from large documents. Without a hierarchical framework, these models can't effectively traverse long texts to retrieve the most relevant sections, often resulting in incomplete or overly broad responses to user queries.

4. **Context Loss**: When handling large texts, many models tend to either condense the information too much, losing important details, or produce repetitive and redundant summaries. This results in a loss of critical context, making it difficult to provide nuanced answers to complex queries.

5. **Scalability Issues**: Many current approaches do not scale well with large datasets or collections of documents. They require extensive computational resources and time to process, making them impractical for applications requiring quick and efficient retrieval.

How RAPTOR Addresses These Gaps:

- **Hierarchical Tree Structure**: RAPTOR builds a tree structure through recursive summarization, allowing for multi-level abstraction. This structure enables the model to efficiently retrieve information at different levels of detail, overcoming the input size limitation.

- **Multi-Level Retrieval**: By organizing text into a hierarchy, RAPTOR can navigate through the document at various levels, providing more precise and contextually relevant responses. This hierarchical retrieval mechanism enhances the model's ability to answer complex queries that require understanding across different sections of a document.

- **Efficient Context Management**: RAPTOR dynamically manages the context by selectively loading relevant parts of the document into the model's processing window, preserving important context while reducing redundancy.

- **Scalability**: The recursive processing and clustering techniques used by RAPTOR make it more scalable, allowing it to handle large documents and datasets more efficiently than traditional models.

RAPTOR’s innovative approach effectively bridges the gaps in current retrieval models, providing a more structured and scalable solution for processing and retrieving information from extensive texts.

OBJECTIVES:

Recursive Summarization: Create a system that breaks down large documents into smaller parts and summarizes them in a tree structure.

Hierarchical Retrieval: Develop a way to search through this tree structure to find information at different levels of detail.

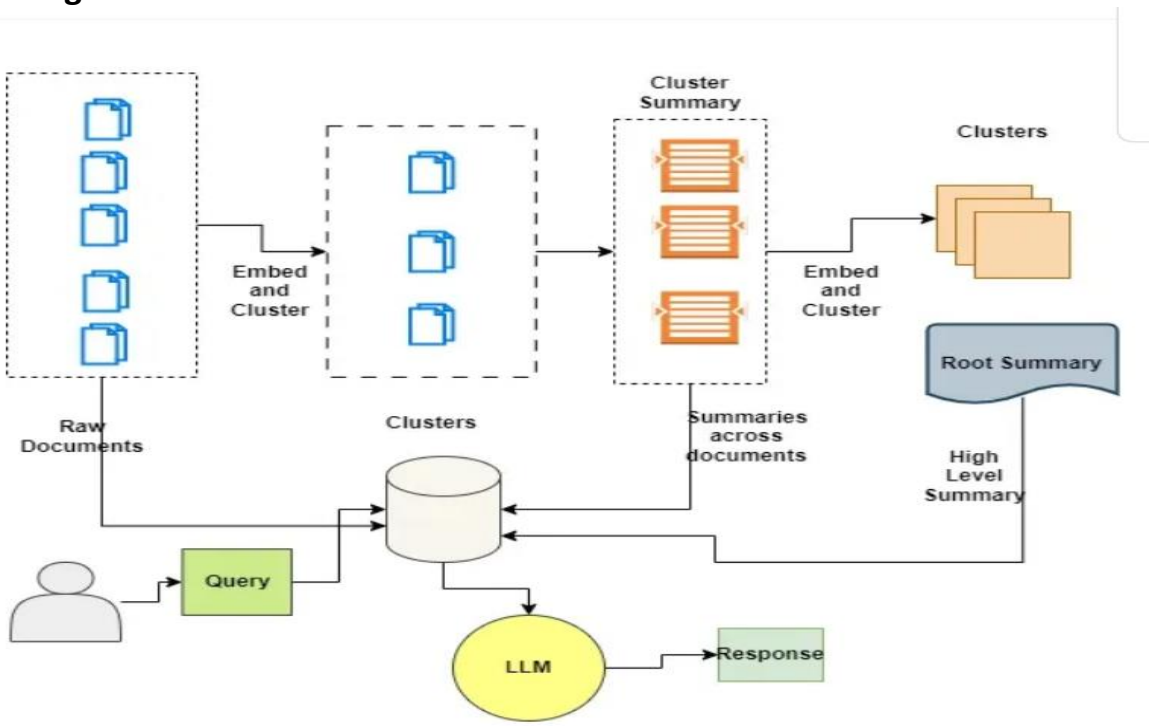
Efficient Context Management: Manage memory usage by dynamically loading relevant information into the model's processing window.

Benchmarking and Evaluation: Test RAPTOR against other models to measure its performance in retrieving information from large documents.

METHODOLOGY:

- **Step 1: Text Chunking**
  Process: Split the large document into smaller chunks using tools like SpaCy or NLTK to keep each chunk meaningful.
- **Step 2: Text Embedding**
  Process: Convert these chunks into numerical vectors using models like SBERT so that they can be processed by the model.
- **Step 3: Clustering**
  Process: Group similar chunks together using a clustering algorithm (like Gaussian Mixture Models) to form clusters that capture main topics in the document.
- **Step 4: Summarization**
  Process: Summarize each cluster using a model like T5 or GPT-3.5 to create a short, concise version of each group of text.
- **Step 5: Tree Construction**
  Process: Organize these summaries into a tree structure, where each node contains a summary and links to more detailed information below it.
- **Step 6: Query Mechanism**
  Process: Implement a system to search through this tree to find the most relevant information for a user's query.
- **Step 7: Evaluation**
  Process: Test RAPTOR on tasks like question answering with large datasets and compare it to other models to see how well it performs.

Block Diagram For RAPTOR:



The image illustrates a process for generating summaries of large datasets of documents. Here's a breakdown of the steps involved:

1. Raw Documents: The process starts with a collection of raw documents (represented by the stack of papers on the left).

2. Embedding and Clustering: Each document is embedded into a numerical representation (often using techniques like word embeddings or transformers). These embeddings are then clustered together based on similarity (represented by the boxes labelled "Embed and Cluster").

3. Cluster Summaries: Summaries are generated for each cluster of similar documents (represented by the boxes labelled "Cluster Summary"). These summaries capture the key themes and information present in the documents within that cluster.

4. Summaries Across Documents: The summaries from each cluster are combined to create a summary that represents the entire dataset (represented by the box labelled "Summaries across documents").

5. High-Level Summary: A high-level summary is generated from the combined summaries, providing a concise overview of the dataset's main points (represented by the box labelled "High-Level Summary").

6. Query and Response: A user can then submit a query related to the dataset. The generated summaries are used to provide a relevant and informative response to the query (represented by the arrow from "Query" to "Response").

7. LLM: A language model (LLM) is likely used to generate the summaries and responses, as indicated by the label "LLM" in the diagram.

Example of How the Model Answers Queries:

Imagine you have a large research document on climate change. A user asks, "What are the causes of global warming?"

1. **Query Input**: "Causes of global warming."
2. **Tree Traversal**: The RAPTOR model navigates the tree structure, first retrieving high-level summaries (e.g., "Human activities and natural events are key contributors to global warming").
3. **Detailed Retrieval**: The system dives into relevant child nodes for more detailed information, such as specific causes like fossil fuel consumption, deforestation, and industrial emissions.
4. **Output**: The user receives a well-organized summary with links to more in-depth information if needed.

TIMELINE:

- Week 1-2: Set up the environment and start implementing text chunking and embedding.
- Week 3-4: Develop clustering and summarization components.
- Week 5-6: Build the hierarchical tree structure and integrate the summarization module.
- Week 7-8: Implement and test the query mechanism.
- Week 9-10: Conduct final tests, optimize the model, and prepare documentation.

EXPECTED OUTCOME:

Implementing RAPTOR is expected to improve how we retrieve information from large documents by using a new method that breaks down and organizes text into a tree structure. This should make it easier to find relevant information more accurately than traditional models, especially in complex documents like research papers or legal texts.

REFERENCES:

1. Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, Christopher D. Manning. "**RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval**" Submitted on 31 Jan 2024. DOI: https://doi.org/10.48550/arXiv.2401.18059

2. Iz Beltagy, Matthew E. Peters, and Arman Cohan. **Longformer: The Long-document Transformer**, 2020. URL: https://arxiv.org/abs/2004.05150. arXiv preprint arXiv:2004.05150

3. Brown, T. B., et al. (2020). *Language Models are Few-Shot Learners*. Link

4. Devlin, J., et al. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Link

5. **Studies Comparing Other Models**:

   - **BERT and GPT-3 Limitations**:
     - Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Link to Paper
     - Brown, T. B., et al. (2020). *Language Models are Few-Shot Learners*. Link to Paper
   - **Retrieval Models**:
     - Karpukhin, V., et al. (2020). *Dense Passage Retrieval for Open-Domain Question Answering*. Link to Paper
     - Yang, P., et al. (2017). *Anserini: Reproducible Ranking Baselines Using Lucene*. Link to Paper
   - **Summarization Models**:
     - Raffel, C., et al. (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. Link to Paper
     - Liu, Y., et al. (2019). *Pre-training with Extracted Gap-sentences for Abstractive Summarization*. Link to Paper

6. **Documentation for Tools and Libraries**:
   - **Sentence-Transformers**:
     - *Sentence-Transformers: Documentation*. Link to Documentation
   - **UMAP**:
     - *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. Link to Documentation

ANNEXURE:

Sarthi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., & Manning, C. D. (2024). Raptor: Recursive abstractive processing for tree-organized retrieval. *arXiv preprint arXiv:2401.18059*.