RAPTOR: RECURSIVE ABSTRACTIVE PROCESSING

FOR TREE-ORGANIZED RETRIEVAL\_v1:

Source Code The source code for RAPTOR will be publicly available in https://github.com/parthsarthi03/raptor.

A diagram of a diagram

Description automatically generated

Step1. Document🡪chunks(100)(to preserve the contextual and semantic coherence, if sentence exceed 100 tokens the whole sentence moves to the next chunk rather than cutting into mid) 🡪 clustered 🡪 summarized(GPT3.5-turbo) 🡪re-embedding(SBERT)

Leafe nodes have two values: chunk and its embedding using SBERT

Step2. Follow step1 until further clustering becomes infeasible, resulting in a structured, multi-layered tree representation of the original documents.

Note: Scalability of both build time and token expenditure is shown below

A graph of a graph of a graph

Description automatically generated with medium confidence

Step3: For querying within the tree, two distinct strategies are used: tree traversal and collapsed tree. The tree traversal method traverses the tree layer-by-layer, pruning and selecting the most relevant nodes at each level. The collapsed tree method evaluates nodes collectively across all layers to find the most relevant ones.

**Clustering Algorithm:**

GMM: It offers both flexibility and a probabilistic framework. where nodes can belong to multiple clusters without requiring a fixed number of clusters. This flexibility is essential because individual text segments often contain information relevant to various topics, thereby warranting their inclusion in multiple summaries.

The high dimensionality of vector embeddings presents a challenge for traditional GMMs, as distance metrics may behave poorly when used to measure similarity in high-dimensional spaces. To mitigate this, we employ Uniform Manifold Approximation and Projection (UMAP), a manifold learning technique for dimensionality reduction . The number of nearest neighbors’ parameter, n neighbors, in UMAP determines the balance between the preservation of local and global structures. Our algorithm varies n neighbors to create a hierarchical clustering structure: it first identifies global clusters and then performs local clustering within these global clusters. This two-step clustering process captures a broad spectrum of relationships among the text data, from broad themes to specific details.

Should a local cluster’s combined context ever exceed the summarization model’s token threshold, our algorithm recursively applies clustering within the cluster, ensuring that the context remains within the token threshold.

In GMM, the number of parameters k is a function of the dimensionality of the input vectors and the number of clusters.

With the optimal number of clusters determined by BIC, the Expectation-Maximization algorithm is then used to estimate the GMM parameters, namely the means, covariances, and mixture weights. While the Gaussian assumption in GMMs may not perfectly align with the nature of text data, which often exhibits a sparse and skewed distribution, our empirical observations suggest that it offers an effective model for our purpose. We run an ablation comparing GMM Clustering with summarizing contiguous chunks and provide details.

**Querying:**

**A diagram of a tree

Description automatically generated**

**Tree traverse method:** method first selects the top-k most relevant root nodes based on their cosine similarity to the query embedding. The children of these selected nodes are considered at the next layer and the top-k nodes are selected from this pool again based on their cosine similarity to the query vector. This process is repeated until we reach the leaf nodes. Finally, the text from all selected nodes is concatenated to form the retrieved context.

1. Start at the root layer of the RAPTOR tree. Compute the cosine similarity between the

query embedding and the embeddings of all nodes present at this initial layer.

1. Choose the top-k nodes based on the highest cosine similarity scores, forming the set S1.
2. Proceed to the child nodes of the elements in set S1. Compute the cosine similarity between

the query vector and the vector embeddings of these child nodes.

1. Select the top k child nodes with the highest cosine similarity scores to the query, forming

the set S2.

1. Continue this process recursively for d layers, producing sets S1, S2, . . . , Sd.
2. Concatenate sets S1 through Sd to assemble the relevant context to the query.

**Collapse tree methos:** It searches relevant information by considering all nodes in the tree simultaneously,

1. First, collapse the entire RAPTOR tree into a single layer. This new set of nodes, denoted

as C, contains nodes from every layer of the original tree.

1. Next, calculate the cosine similarity between the query embedding and the embeddings of

all nodes present in the collapsed set C.

1. 3. Finally, pick the top-k nodes that have the highest cosine similarity scores with the query.

Keep adding nodes to the result set until you reach a predefined maximum number

****

Figure 3: Comparison of querying methods.

Results on 20 stories from the QASPER dataset

using tree traversal with different top-k values,

and collapsed tree with different context lengths.

Collapsed tree with 2000 tokens produces the best

results, so we use this querying strategy for main results.

**CONCLUSION**

RAPTOR is a novel tree-based retrieval system that augments the parametric knowledge of large language models with contextual information at various levels of abstraction. By employing recursive clustering and summarization techniques, RAPTOR creates a hierarchical tree structure that is capable of synthesizing information across various sections of the retrieval corpora. During the query phase, RAPTOR leverages this tree structure for more effective retrieval. RAPTOR not only outperforms traditional retrieval methods but also sets new performance benchmarks on several question-answering tasks.