SiReRAG: Indexing Similar and Related Information for Multihop Reasoning

Nan Zhang, Prafulla Kumar Choubey, Alexander Fabbri, Gabriel Bernadett-Shapiro, Rui Zhang, Prasenjit Mitra, Caiming Xiong, Chien-Sheng Wu

ICLR 2025

Modular RAG

- Retrieval-augmented generation (RAG) augments large language models (LLMs) with highly specialized and constantly updated knowledge.

- A typical RAG may involve:
  - Chunking
  - Embedding
  - Indexing
  - Retrieval
  - Reranking
  - LLM response generation

RAG Indexing

- RAG indexing focuses on organizing a large amount of data and serves as an upstream step of retrieval.

- However, none of the existing methods address the importance of indexing from both similarity and relatedness sides. → limits a holistic understanding of the provided dataset
    - Similarity: semantic distance of text pieces
    - Relatedness: the degree of connection of texts based on signals such as entities and propositions

Existing RAG Indexing Methods for Multihop

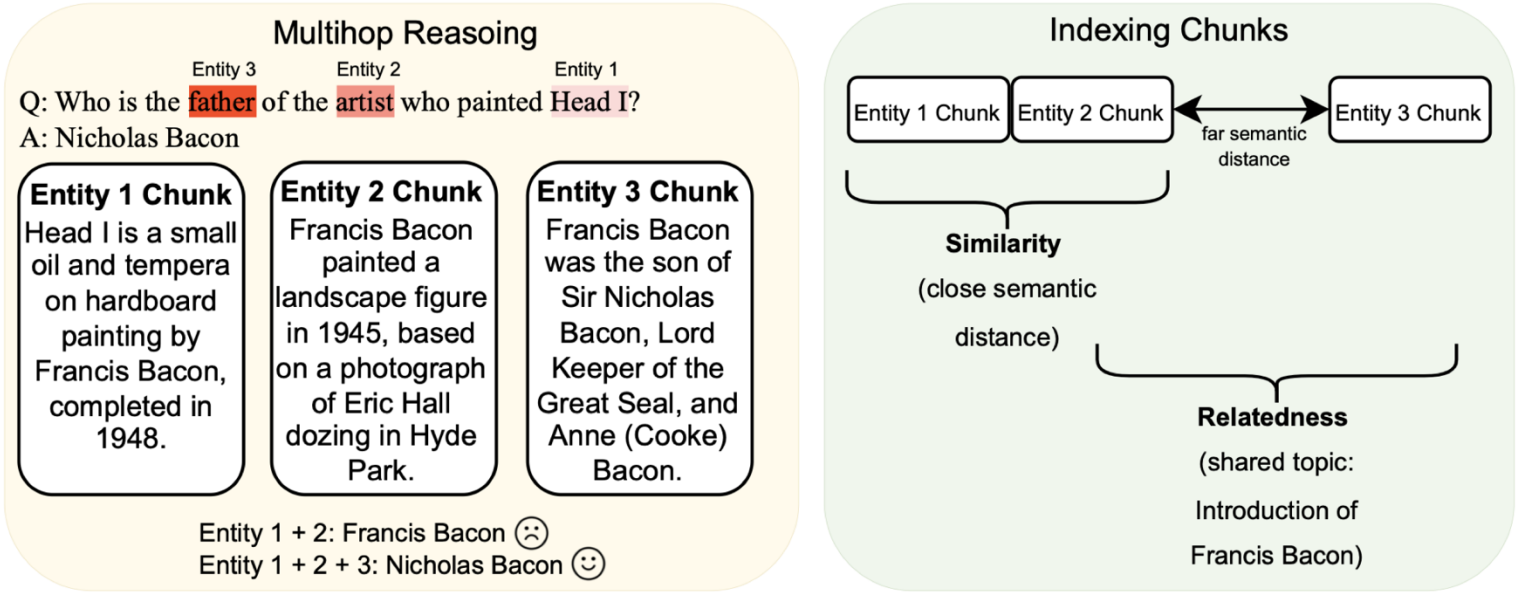


Figure 1: **Challenges of existing RAG indexing methods for multihop reasoning.** Entity 1 and 2 chunks contain similar information while entity 2 and 3 chunks contain related contents. Since synthesizing information only based on entity 1 and 2 (or entity 2 and 3) will lead to a higher probability of a wrong answer, an indexing method that considers both similarity and relatedness is needed to maximize retrieving relevant knowledge for multihop questions.

Bottleneck of Solely Modeling Similarity or Relatedness

- We perform two kinds of clustering philosophies (either **similarity** or **relatedness**) on deep representations of MuSiQue chunks.
- Clustering results based on similarity or relatedness are not identical, and we can leverage both to facilitate a more comprehensive knowledge integration process.

Table 1: Coverage percentage between different clusters. MuSiQue clusters include supporting and distractor passages. The "all" setting treats both as gold, while "supporting only" uses only supporting passages as gold. We first show the coverage of supporting or all passages under two clustering philosophies. We then report the overlapping ratio between "supporting only" similarity and relatedness to motivate our work of combining both philosophies.

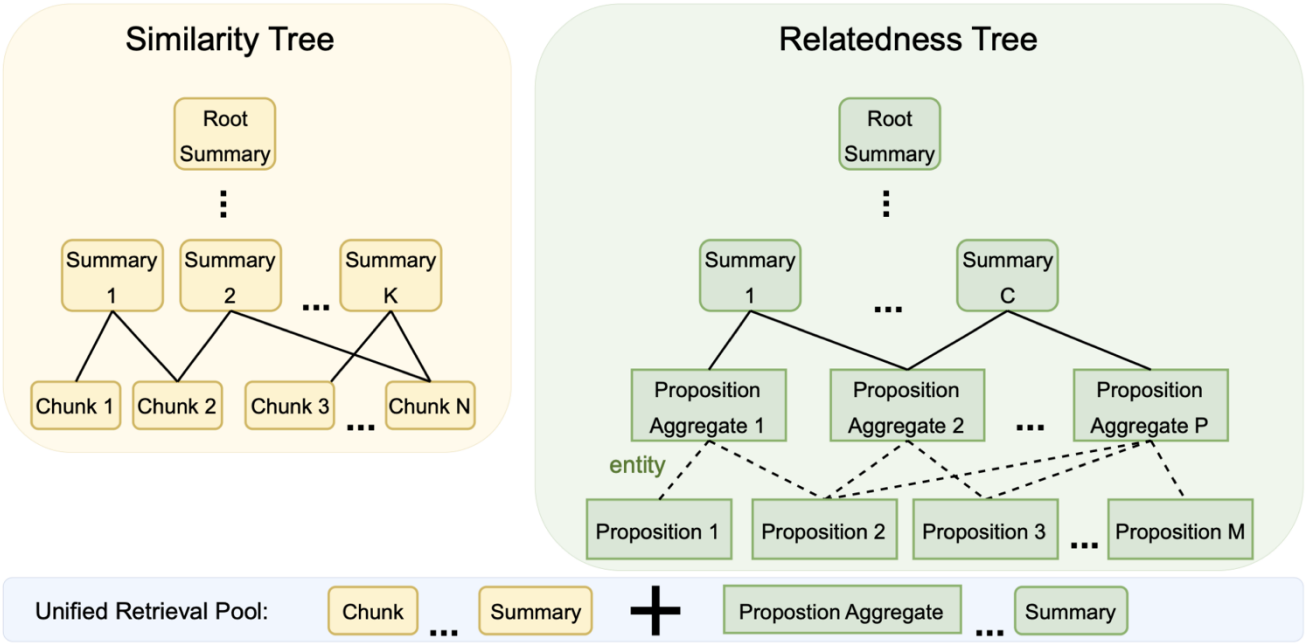
	Similarity Coverage		Relatedness Coverage		Overlapping Ratio	
	Supporting Only	All	Supporting Only	All	Overlap@Similarity	Overlap@Relatedness
Coverage	19.14%	10.70%	13.94%	8.51%	50.15%	68.85%

Exploring A Hierarchical Structure of Text Chunks

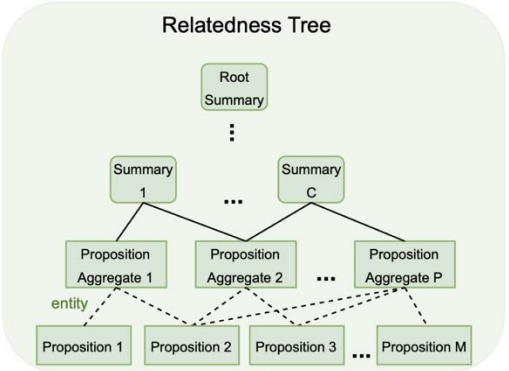
- Providing multiple levels of data abstraction, we find that a hierarchical structure of text chunks does not offer improvement.
  - Prompt LLMs to identify a two-level hierarchy for chunks: low and high abstractiveness.
  - Recursive summaries (for similar tree nodes) start from the second level (the level above the bottom level).
  - Thus, the second level will contain both summary nodes (of the low-abstractive chunks) and high-abstractive chunks.
  - As a different design against placing all chunks at the bottom, this hierarchy of chunks affects summary nodes due to the differences of their children.

SiReRAG

- We propose SiReRAG (RAG indexing of **si**milarity and **re**latedness).



SiReRAG



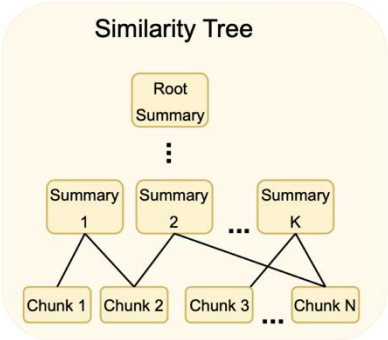
- **Relatedness** side: synthesize information based on extracted entity
  - Proposition: a factual statement from a chunk
  - Entity and proposition extraction
    1. Rewrite chunks of a larger corpus using LLaMA-70B
    2. Extract propositions and their associated entities using LLaMA-70B
    3. Consolidate all extracted propositions and entities to fine-tune a smaller Mistral model.
    4. Fine-tuned models is used to extract propositions and their associated entities

GPT-4o generates similar outputs.

  - Concatenate related propositions that share the same entity to form proposition aggregates.
  - Proposition aggregates and recursive summaries are used for constructing relatedness tree.

SiReRAG

- **Similarity** side: synthesize information based on semantic similarity
  - We adopt RAPTOR tree that has text chunks at the bottom and recursive summaries (for similar tree nodes) on top.



- SiReRAG: indexing both **similarity** and **relatedness**
  - Similarity and relatedness trees are constructed independently for efficiency.
  - SiReRAG flattens all nodes from both trees into a unified retrieval pool.

Overall Results

Table 4: QA performance of SiRERAG and baselines. As elaborated in Section 5.4, GPT-4o is used to handle QA for all models, and we use two different LLMs (specified in the parentheses) to build indexing structures. We highlight the best scores using either LLM for indexing in green color.

	MuSiQue		2Wiki		HotpotQA		Average	
Model	EM	F1	EM	F1	EM	F1	EM	F1
HippoRAG (GPT-3.5-Turbo)	32.60	43.78	66.40	74.01	59.90	74.29	52.97	64.03
RAPTOR (GPT-3.5-Turbo)	35.30	47.47	54.90	61.20	58.10	72.48	49.43	60.38
GraphRAG (GPT-4o)	12.10	20.22	22.50	27.49	31.70	42.74	22.10	30.15
RAPTOR (GPT-4o)	36.40	49.09	53.80	61.45	58.00	73.08	49.40	61.21
SiRERAG (GPT-3.5-Turbo)	38.90	52.08	60.40	68.20	62.50	77.36	53.93	65.88
SiRERAG (GPT-4o)	40.50	53.08	59.60	67.94	61.70	76.48	53.93	65.83

RAPTOR: similarity only

HippoRAG: relatedness only

GraphRAG: relatedness only (more for query-focused summarization of an entire corpus)

SiReRAG Applicability

Table 6: Applicability of SiReRAG when a specific retrieval method is selected. We feed our non-indexing models with the retrieval pool of SiReRAG and see whether QA performance improves.

	MuSiQue		2Wiki		HotpotQA		Average	
Variants	EM	F1	EM	F1	EM	F1	EM	F1
BM25	25.90	35.88	53.00	58.58	57.70	71.32	45.53	55.26
RAPTOR + BM25	27.00	38.91	50.60	57.00	56.90	70.88	44.83	55.60
SiReRAG + BM25	35.00	47.66	58.20	65.72	61.70	75.88	51.63	63.09
ColBERTv2	34.00	46.80	52.90	59.48	59.00	73.42	48.63	59.90
RAPTOR + ColBERTv2	34.20	47.17	50.30	57.51	57.90	72.31	47.47	59.00
SiReRAG + ColBERTv2	38.10	51.32	56.70	64.74	60.90	75.72	51.90	63.93

SiReRAG significantly improves other retrieval methods (e.g., reranking).

Conclusion

- SiReRAG indexes both similarity and relatedness for more comprehensive knowledge synthesis, which addresses the bottleneck of solely indexing one signal on multihop reasoning benchmarks.

- SiReRAG can potentially improve other non-indexing methods under RAG setup.

- Excited to see how people extend the idea of SiReRAG!

Questions?

- Paper: https://arxiv.org/abs/2412.06206
- Code: https://github.com/SalesforceAIResearch/SiReRAG
- Email: njz5124@psu.edu
- Homepage: https://zn1010.github.io