Lil Lisa v2 Support Chatbot

Carlos Escobar

Introduction

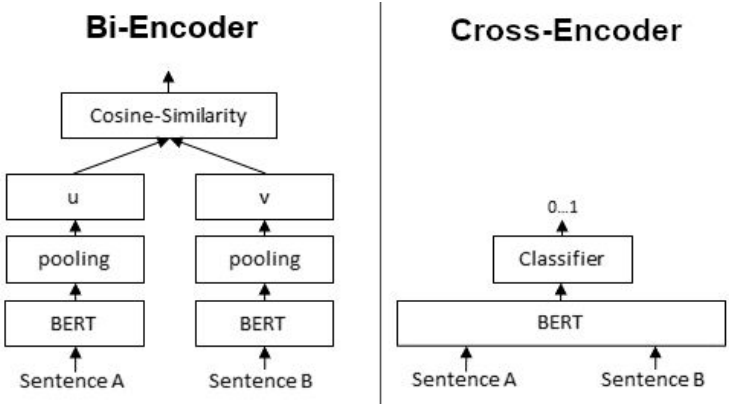
- One critical aspect in enhancing the functionality of LLMs lies in their ability to access and integrate information from external sources

- To address this limitation with Lil Lisa v2, I created a Retrieval-Augmented Generation (RAG) system with the capability to query a database, enhancing answers to a given query from a user

- In this study, I explored several chunking and indexing methods, aiming to evaluate its impact on response quality and efficiency

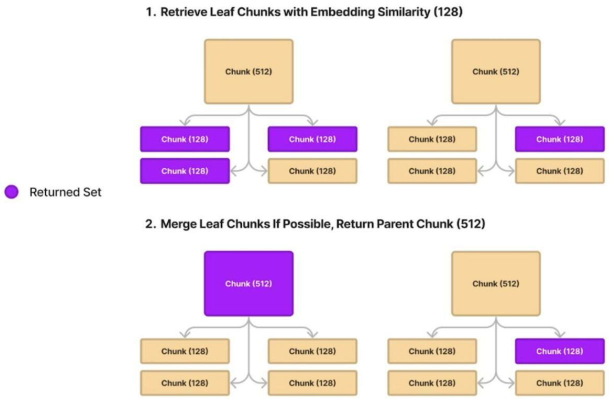
Background

- Vector Search
  - Computes the similarity between words and phrases by computing the difference of vectors in a space
- Keyword Search
  - Relies on matching exact words between phrases to compute relevancy
- Bi-Encoders vs. Cross-Encoders
  - Cross-Encoders are more accurate, but slower and not meant for large datasets because of the high computational intensity required



Things I tested

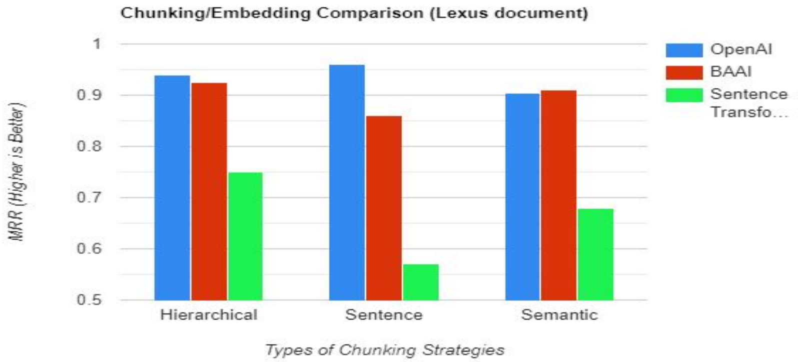
[FOUND ONLINE] *Diagram of how leaf nodes are merged*

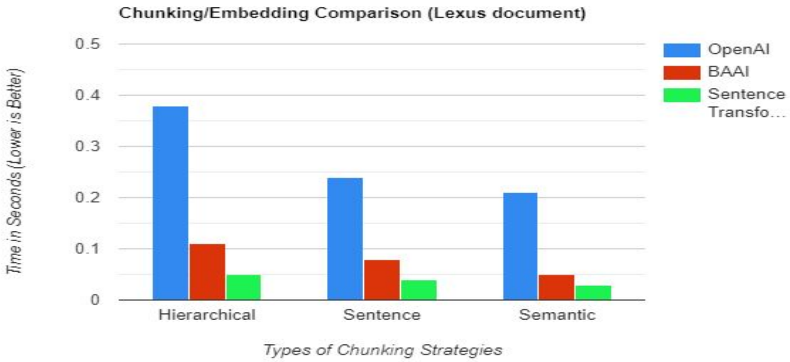


- Chunking Methods
  - Hierarchical
  - Sentence
  - Semantic

- Embedding Models
  - OpenAI's text-embedding-3-large (3072 dimensions)
  - BAAI's bge-large-en-v1.5 (1024 dimensions)
  - Sentence Transformers' all-MiniLM-L12-v2 (384 dimensions)

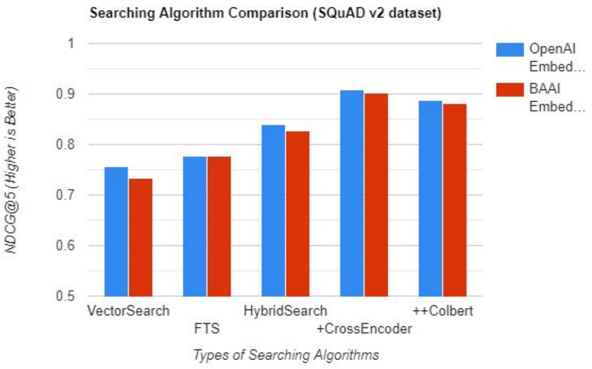
Comparison of Chunking Strategies Across Embedding Models (Lexus Document)

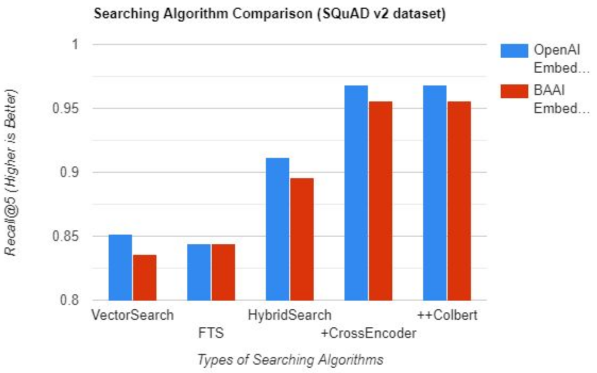


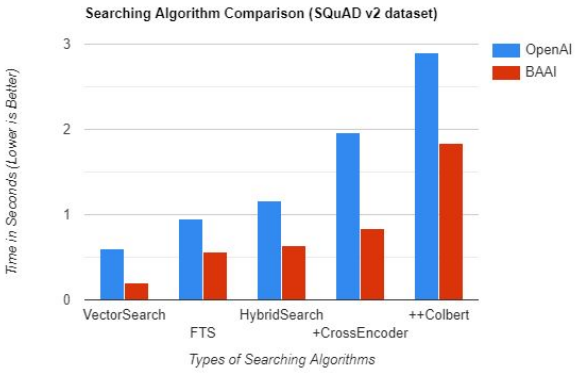


- Key Findings:
  - OpenAI and BAAI embeddings are superior
  - Semantic chunking requires too much preprocessing with less reliable outcomes
- Conclusion:
  - Top Variations:
    - OpenAI Hierarchical
    - BAAI Hierarchical
    - OpenAI Sentence
- Next Steps: Discontinue semantic chunking for larger sets of documents

Comparison of Searching Strategies Across Embedding Models (SQuAD v2 dataset)

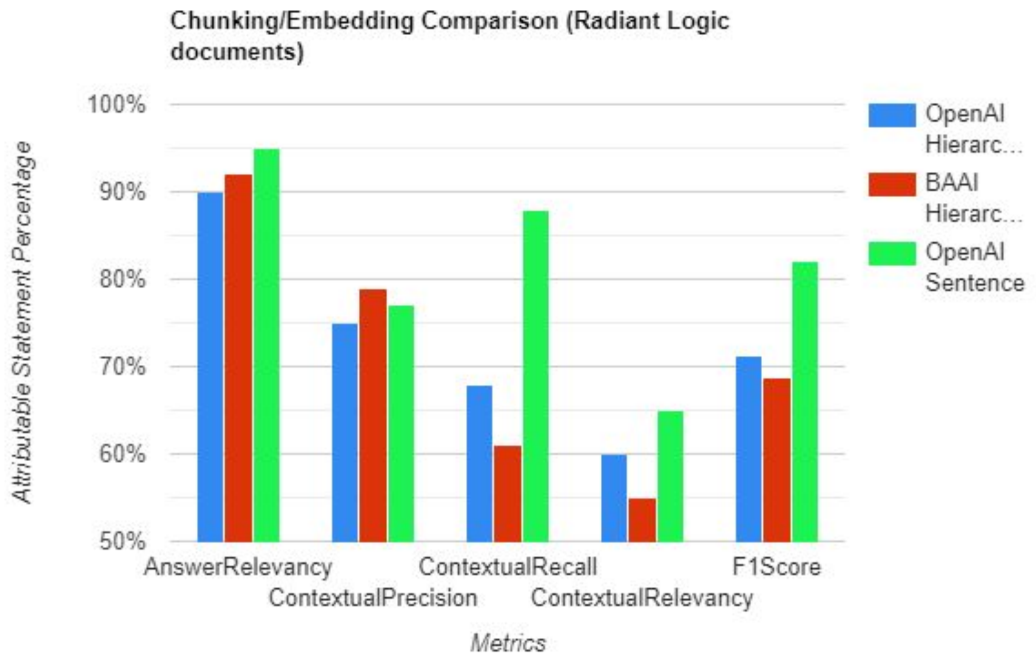






- Key Findings:
  - Slightly better retrieval with OpenAI embeddings for an increased time-per-query
  - Colbert seems to worsen results
- Conclusion:
  - Next Steps: Try another dataset

Comparison of Chunking Strategies Across Embedding Models (Radiant Logic documents)



- Key Findings:
  - OpenAI Sentence is considerably better than the other two
  - OpenAI Hierarchical is marginally better than BAAI Hierarchical
- Conclusion:
  - Next Steps: Find the best reranking strategy

Comparison of Chunking and Indexing Strategies (Radiant Logic documents)

Method	Average Rank	MRR	Recall@5	Recall@10	Recall@20	NDCG@5	NDCG@10	NDCG@20	Time Per Query (s)
Chunking Variation 1 + Hybrid Search	3.865	0.622	0.838	0.865	1.000	0.668	0.677	0.710	**1.020**
Chunking Variation 1 + Hybrid Search + Cross-Encoder	**1.703**	**0.858**	**0.973**	0.973	1.000	**0.885**	**0.885**	**0.892**	**2.034**
Chunking Variation 1 + Hybrid Search + ColBERT v2.0	**1.892**	0.765	0.946	**1.000**	1.000	0.805	0.823	0.823	10.293
Chunking Variation 1 + Hybrid Search + Cross-Encoder + ColBERT v2.0	**1.865**	0.786	**0.973**	0.973	0.973	0.832	0.832	0.839	4.018
Chunking Variation 2 + Hybrid Search	4.162	0.570	0.865	0.892	0.946	0.637	0.646	0.661	**1.200**
Chunking Variation 2 + Hybrid Search + Cross-Encoder	1.946	**0.858**	0.946	0.946	1.000	**0.877**	**0.877**	**0.891**	2.466
Chunking Variation 2 + Hybrid Search + ColBERT v2.0	2.378	0.793	0.919	0.973	0.973	0.818	0.836	0.836	13.738
Chunking Variation 2 + Hybrid Search + Cross-Encoder + ColBERT v2.0	2.135	**0.814**	0.946	0.946	1.000	**0.844**	**0.844**	**0.858**	4.61

Entries highlighted in **green** indicate the best performance, entries highlighted in **blue** indicate the second best performance and entries highlighted in **red** indicate the third best performance.

- Top Variations (in order):
  1) Chunking Variation 1 + Hybrid Search + Cross-Encoder
  2) Chunking Variation 2 + Hybrid Search + Cross-Encoder
  3) Chunking Variation 2 + Hybrid Search + Cross-Encoder + ColBERT v2.0

Comparison of Chunking and Indexing Strategies (Radiant Logic documents)

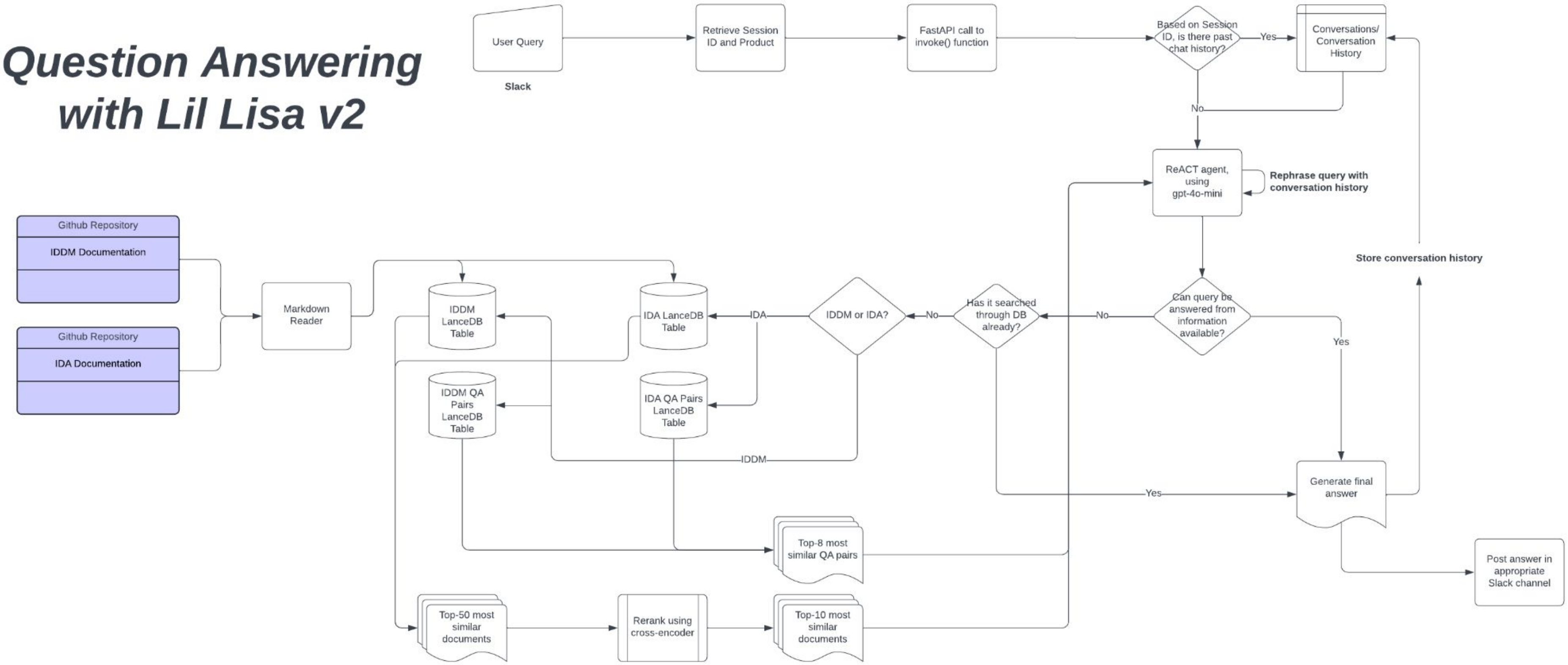
Method	Answer Correctness	Contextual Precision	Contextual Recall	F1 Score	Time Per Query (s)
Chunking Variation 1 + Hybrid Search + Cross-Encoder	**0.877**	0.971	**0.944**	**0.948**	**3.19**
Chunking Variation 2 + Hybrid Search + Cross-Encoder	0.822	**0.983**	0.899	0.905	3.440
Chunking Variation 2 + Hybrid Search + Cross-Encoder + ColBERT v2.0	0.849	0.961	0.918	0.920	4.468

Entries highlighted in green indicate the best performance.

- Conclusion:
  - ‘Chunking Variation 1 + Hybrid Search + Cross-Encoder’ had the best overall performance
  - OpenAI’s text-embedding-3-large 3072 dimensions were sufficient in retaining the semantic meaning of large chunks

Architecture/Diagram

Question Answering with Lil Lisa v2



Improvements over Lil Lisa v1

- No need to tag the bot when you want an answer to your question (unless 2+ people in conversation)
- Multi-turn conversation
- If relevant context doesn't exist in the documentation, the agent is capable of telling a user the question is out of scope
- An "Expert" is invited into the conversation if user gives an "SOS" reaction
- Bot can be integrated outside of Slack, such as web portals
- We now have Lil Elvis! (for IDA queries)
- "/" commands have been streamlined making it easy for "expert" to review and update QA pairs accordingly
- Knowledge base is connected to active github repositories, allowing the "/rebuild_docs" method to access up-to-date information

Future Enhancements

- Version-specific answers (coming very soon)
- Allow users to provide screenshots with their questions
- Return screenshots/images with answers
- Response streaming
- Let users Direct Message the bot for Question-Answering
- Refine the handling of expert answers in answer synthesis

Appendix

LlamaIndex overview: https://docs.llamaindex.ai/en/stable/
LanceDB start guide: https://lancedb.github.io/lancedb/basic/
DeepEval start guide: https://docs.confident-ai.com/docs/getting-started
Ragas overview: https://docs.ragas.io/en/latest/index.html
BERT Paper: https://arxiv.org/abs/1810.04805
ReACT Paper: https://arxiv.org/abs/2210.03629

Github repository:
https://medium.com/@carlos-a-escobar/deep-dive-into-the-best-chunking-indexing-method-for-rag-5921d29f138f
Medium.com blog:
https://medium.com/@carlos-a-escobar/deep-dive-into-the-best-chunking-indexing-method-for-rag-5921d29f138f