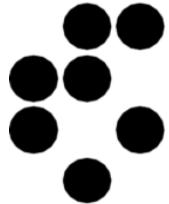


Implementing contextual retrieval in RAG pipeline



Jožef Stefan Institute



Lan Alexander Rojs, E3 department

Contextual retrieval - preprocessing text

- We have a document in .txt format and we would like to process it for future chunking
- `process_text.py`:
 - normalizes whitespace and punctuation,
 - removes non UTF-8 characters,
 - handles list markers like (1) or (a)...

Contextual retrieval - chunking

- Once we have cleaned the document, we would like to parse it into smaller chunks that are more optimal for storage and future retrieval
- Different chunking strategies:
 - **fixed-size chunking**
 - Content-aware chunking
 - naive chunking (splitting by sentence)
 - Document structure-based chunking

Contextual retrieval - chunking

- We store chunks in a JSON file and give them unique ids so we can trace everything back
- Parameters I used in chunks.py:
 - Tokens per chunk: 300
 - Overlap between chunks in tokens: 50
 - Tokenizer: BAAI/bge-small-en-v1.5

Contextual retrieval - embedding

- Once the document is chunked, we embed each chunk (assign a vector to it) and store embeddings + metadata in an SQL database - useful for semantic and lexical retrieval
- Parameters I used in embedding.py:
 - Embedding model: BAAI/bge-small-en-v1.5
 - Batch size: 64
 - L2 normalization: True

Contextual retrieval - contextualizing

- For each text chunk we want to generate a short 2-3 sentence contextualized summary that explains what that passage says and how it functions in the document.
- We process chunks sequentially: build a temporary database that stores information about all contextualized chunks so far

Contextual retrieval - contextualizing

- For each chunk:
 - Collect neighbouring raw context.
 - Retrieve relevant previous contextualized summaries.
 - Compose a structured LLM prompt with these contexts and the current passage and call LLM to produce a short self-contained summary.
 - Clean and store it in the retrieval index for future chunks.

Contextual retrieval - contextualizing

- Parameters I used in contextualize.py:
 - LLM model: llama3:8b, gemma3:4b-it-qat
 - Temperature: 0.0
 - Before and after neighbours: 2 + 2
 - Max. retrieved chunks to include as context: 3
 - Contextualizing prompt: modified Anthropic contextualizing prompt (stricter, more specific instructions to prevent hallucinations due to smaller LLM models)

Contextual retrieval - extracting claims

- We pass both the original chunk and its contextualization to the LLM to identify explicit statements and extract self-contained, factual claims
- The claims are stored in a JSON file together with a source quote (the most specific quote from the original chunk that supports the claim)
- Parameters used in `extract_claims.py`:
 - LLM model: llama3:8b
 - Temperature: 0.0
 - Extracting prompt: modified GraphRAG prompt for extracting claims, but stricter and more specific

Possible improvements

- Different chunking strategies
- Larger LLM models, try some other embedding models
- Experiment with prompts used for contextualizing and extracting claims
- Experiment with LLM temperature...?