Rocket Club AI
Final Report Findings

Team 558 • 2025-04-02

Agenda

- Chunking
- Metadata
- Hybrid Search
- Advanced RAG Techniques
- Result Reranking
- Q&A

Chunking

Problem Statement

- Currently using dynamic chunking within sections defined by OCR engine
    - Large chunks being cited for small pieces of information, human verifiers search too long
    - Chunks sometimes not large enough to capture context for summary questions
- New system should properly chunk information based on what questions will be asked to find it
    - Factual information should be embedded in small chunks
    - Information which would be asked for in a summary question should be embedded in large chunks with plenty of metadata

Factors Influencing Chunking Method

- LLM Context Length
    - Sweet spot–provide enough context but prevent hallucinations
- Text Structure
    - Different kinds of text require different chunking methods
- Embedding Model
- Type of Questions
    - Different kinds of questions might be answered better with different chunking methods
- Cost and Speed Considerations

Types of Chunking

- Fixed-Size Chunking
  - Set each chunk to a fixed size with a little overlap
- Variable-Size Chunking
  - Use a program to chunk sections by semantic elements like punctuation
- Content Aware Chunking
  - Use an LLM to chunk sections based on similar semantic meaning
- Hybrid Chunking
  - Use multiple of the above methods over the same or different parts of the document
- Using Metadata
  - Content Enriched Chunking
  - Subdocument Chunking

Mix of Granularity Chunking Method

- Splits data into chunks based on questions asked to retrieve certain data using a router
  - Factual information given smaller chunks, summary question information given larger chunks
- Mix of Granularity Graph chunking allows context from multiple documents or non-adjacent places in the same document to be chunked together
- MoG authors recommend addition of metadata to some chunks
  - Summaries of previous or similar sections should be prepended to summary question chunks
- Code is public on GitHub

Fine Tuning models with metadata

Problem

The LLM chatbot gives broad answers to queries. Thus when typing for example “rent” into the chatbot

Goals:

- Fine tuning of models such with metadata.
- Automate metadata extraction

Types of metadata

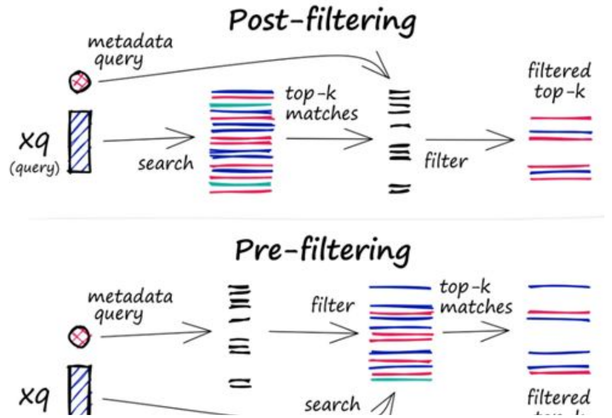
- Descriptive metadata (title, keywords, author’s name)
- Structural metadata (tables, order of the document)
- Administrative metadata (Access control, copyright information)

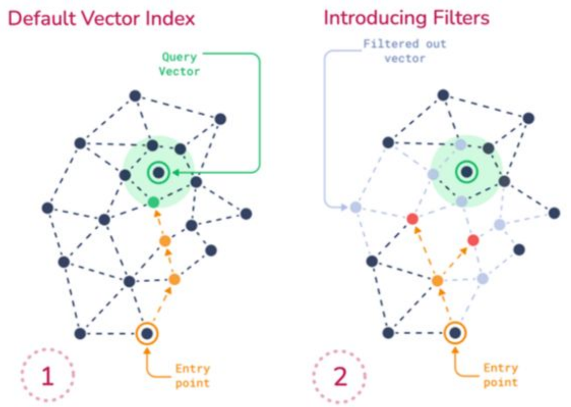
RAG implementation

- Relevance ranking: documents with matching metadata are given a higher priority during the retrieval.
- Enhanced Query Understanding: Metadata adds additional context to Queries
- Metadata can be embedded to find matches beyond a document’s content.

Metadata filtering

- Post Filtering does metadata filtering after a vector search
- Pre Filtering does metadata filtering before a vector search
- Fuzzy filtering is utilized to enable the usage of metadata in models by correcting typos and errors





Metadata frameworks

- Haystack:
- Haystack has a custom component QueryMetadataExtractor. This can be automated through a pipeline.
- Haystack metadata enrichment

Deasy Labs:

- AI suggested metadata tags
- Directly connect metadata to vector base
- Automate metadata updates

Key recommendations

1. Incorporate Deasy Labs to automate metadata management
2. Utilize pre and post filtering
3. Implement RAG during pre processing, retrieval and ranking

Hybrid Search Models For Lease Abstraction

Challenges With Lease Abstraction and Existing Search Models

Complex Legal Language

- Lease documents often contain intricate terminology and varied clause structures that require precise interpretation.

Inconsistent Formatting

- Different landlords, legal teams, and jurisdictions format leases differently, making standardization difficult.

Synonym Variations

- Organizations must process thousands of leases, requiring scalable and efficient search models.

Volume of Documents

- Similar clauses may be written in multiple ways, making keyword-based retrieval less effective.

Differences Between Sparse and Dense Embedding

Sparse Embedding

- Keyword-based retrieval
- Efficient, interpretable, but lacks semantic understanding.

Dense Embedding

- Context-aware retrieval
- Captures meaning beyond exact keyword matches.

Overview of Sparse Embedding Models

Okapi BM25

- ➢ Ranks documents based on term frequency and inverse document frequency.
- ➢ Strong for keyword-based retrieval.

Query Likelihood (QL)

- ➢ Probabilistic ranking approach.
- ➢ Uses Maximum Likelihood Estimation (MLE) to rank documents.
- ➢ Effective for keyword retrieval in structured legal documents.

Overview of Dense Embedding Models

Sentence-Bert (SBERT)

- ➢ Captures sentence-level meaning.
- ➢ Effective for lease clause similarity matching.

Microsoft's MPNet

- ➢ Stronger contextual awareness.
- ➢ Outperforms standard BERT in semantic tasks.

Hybrid Solution

Combining Sparse and Dense Methods:

- ➢ A hybrid model integrates the precision of sparse algorithms (like BM25) with the semantic depth of dense models(like SBERT and MPNet).
- ➢ **BM25**: Filters documents based on keyword frequency, ensuring exact match relevance.
- ➢ **SBERT** & **MPNet**: Analyze sentence-level meaning and context, ranking results based on semantic similarity

Why Hybrid Works for Lease Abstraction:

Legal language often combines strict terminology with complex phrasing, making it hard for one model type to perform well alone.

Hybrid models can retrieve clauses written in different styles but with similar legal meanings.

What It Looks Like

Metadata

➢ Metadata fields (e.g., clause type, lease date)

Initial Filtering Semantic

➢ BM25 quickly narrows down candidate documents using keyword matching

Semantic Re-Ranking

➢ SBERT or MPNet re-ranks the top documents by evaluating contextual similarity to the search query

Final Recommendations

1. **Adopt a hybrid model** leveraging sparse and dense embeddings
2. **Fine-tune dense models** for domain-specific legal text
3. **Continuously evaluate and optimize** search accuracy

Research Advance Retrieval-Augmented Generation (RAG) Techniques and Identify Best Practices

Problem

AI retrieves relevant text without reasoning through lease clauses, such as identifying exceptions, dependencies, or conflicting terms.

Different leases define similar terms differently such as "base rent" vs. "minimum rent", this overlap confuses the system which can cause redundant or conflicting information in responses.

Goal: Recommend advance RAG techniques which increases:

- Retrieval accuracy
- Contextual awareness
- Understanding of complex queries

GraphRAG

Integrates knowledge graphs to enhance performance of LLMs' retrieval and generation of answers of queries.

- Addresses the RAG system's struggle with maintaining context within long documents such as leases.

Knowledge Graph vs Vector Database

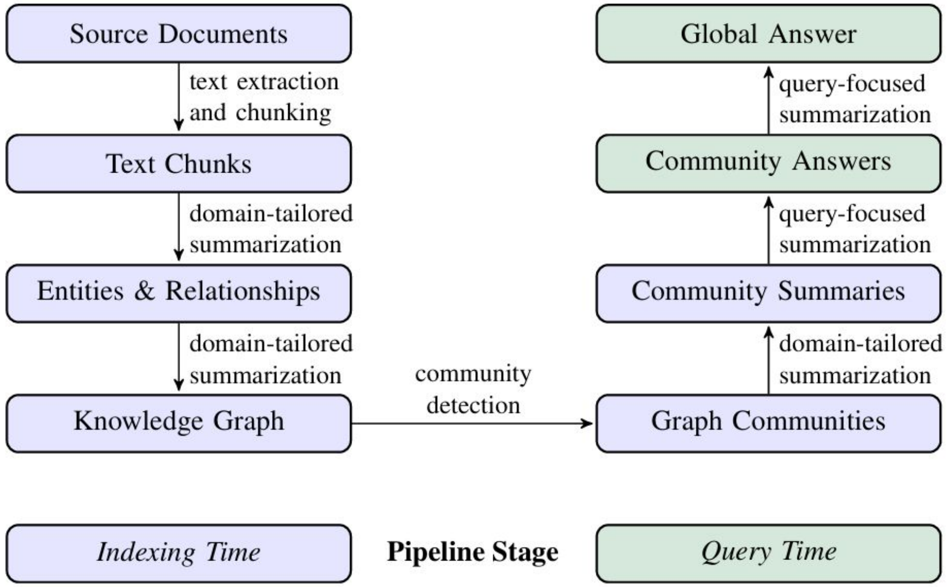
- Vector databases store query and data as numerical vectors by converting data into embeddings,
    - Enable efficient retrieval of unstructured data like text, audio, and images.
- Graph database stores data as nodes and edges
    - Perfect for managing and querying data with intricate relationships

Overview of GraphRAG

Knowledge Graph Construction – Documents are broken into smaller chunks, converted into nodes, and linked based on semantic similarity.

Query Processing – A graph traversal algorithm finds the most relevant information and refines the response using AI.

Visualization – The knowledge graph provides transparency by showing relationships and connection strengths.

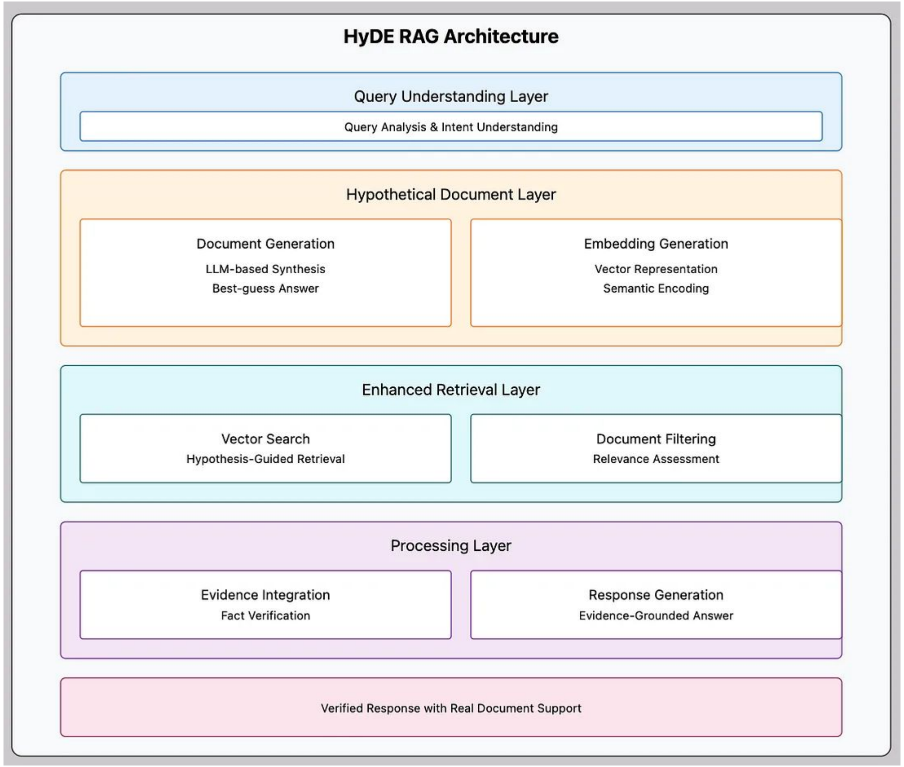


HyDE (Hypothetical Dense Embedding)

Traditional retrieval methods struggle with the semantic gap between short queries and long documents. HyDE bridges this gap by:

- Expanding queries into full hypothetical documents.
- Using these expanded queries to retrieve more contextually relevant documents.
- Improving semantic matching, particularly in legal texts with complex terminology.

By incorporating HyDE, we enhance AI's ability to interpret and retrieve lease terms accurately.



RAPTOR (Recursive Abstractive Processing for Tree Organized Retrieval)

Structures documents into a hierarchical tree to improve context retention and reasoning in long documents. This approach is beneficial for long documents as it:

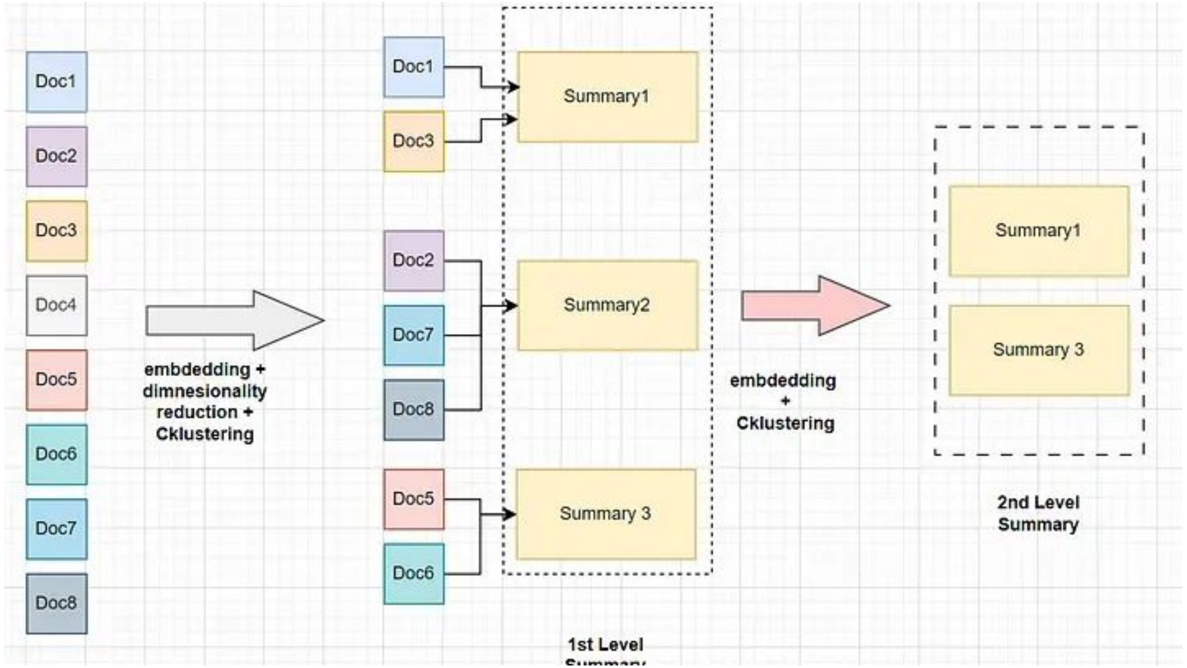
- Groups related content into clusters and summarizes them.
- Builds a hierarchical structure where high-level summaries guide retrieval.
- Optimizes retrieval speed and AI reasoning by reducing redundancy.

For lease abstraction, RAPTOR allows multi-step queries to be handled more efficiently, improving AI's ability to process large lease portfolios.

RAPTOR technique

RAPTOR Tree Construction:

- Chunk & embed raw documents (leaves).
- Cluster similar embeddings.
- Summarize clusters into higher-level nodes.
- Repeat (re-embed, compress, cluster, summarize) until hitting LLM token limits



Recommendations

By combining these techniques, we can improve retrieval accuracy, efficiency, and overall AI performance in lease abstraction.

- Integrating a combination of HyDE query expansion and GraphRAG knowledge graph structured retrieval.
- RAPTOR hierarchical tree system

Enhancing Document Search with Result Reranking

Problem

Embedding models are widely used for similarity search but don’t always return the best results.

Goal: Improve ranking of search results using additional strategies like:

- Keyword matching
- Semantic similarity
- Hybrid ranking models

Reranking Strategies Overview

1. Lexical Matching (BM25) – Keyword relevance
2. Cross-Encoders (ColBERT) – Deep contextual ranking
3. BERT-Based Models – Fine-tuned semantic ranking
4. Query Expansion – Synonyms & paraphrases for better retrieval
5. Diversity Reranking – Avoid redundant results
6. Multi-Modal Fusion – Text & image embeddings
7. Relevance Feedback – Learn from user interactions

Hybrid Reranking with BM25 & Embeddings

- BM25 prioritizes keyword frequency but lacks semantic understanding.
- Embedding Similarity captures meaning but can retrieve unrelated matches.
- Hybrid Approach: Combines BM25 for initial ranking and embeddings for semantic refinement.

Cross-Encoders for Better Ranking

- ColBERT & Cross-Encoders refine ranking by directly comparing query-document pairs.
- More accurate than cosine similarity-based embeddings.
- Computationally expensive but improves ranking precision.

BERT-based Models for Contextual Understanding

- BERT-based models generate context-aware embeddings.
- Fine-tuning on domain-specific datasets improves ranking relevance.
- ColBERT optimizes token-level similarity evaluation.

Query Expansion for Improved Search Accuracy

- Expands queries with synonyms & paraphrases to improve recall.
- Bridges vocabulary gaps between user queries and stored documents.
- LLMs assist in generating semantically similar queries.

Diversity Reranking Multimodal Fusion

- Diversity Reranking prevents excessive clustering of similar results.
- Multi-Modal Fusion integrates text and images for a richer search experience.

Relevance Feedback & Continuous Improvement

- Uses user interactions (clicks, time spent) to refine ranking.
- Dynamic adjustments improve ranking accuracy over time.

Evaluating Reranking Performance

- NDCG (Normalized Discounted Cumulative Gain): Prioritizes high-relevance results.
- MRR (Mean Reciprocal Rank): Measures ranking of the first relevant document.
- Precision at K (P@K): Evaluates how many top-K results are relevant.
- Recall at K (R@K): Measures how well relevant results are retrieved.
- MAP (Mean Average Precision): Averages precision across multiple queries.

Key Recommendation

- Hybrid Reranking: Use BM25 + embeddings.
- Fine-Tuning for Lease Data: Train models on legal documents.
- Query Expansion: Improve recall with LLM-generated synonyms.
- Relevance Feedback: Use user behavior for dynamic ranking.
- Diversity Reranking: Ensure search results are varied.
- Multi-Modal Fusion: Integrate textual and visual elements.

Results

1. Enhanced ranking accuracy for document search.
2. Improved efficiency for real estate & legal professionals.
3. Continuous improvement through feedback & fine-tuning.

Q&A

1. What are some benchmarks or results on response accuracy when using the full context window?
2. What is the recommended range for the size of the metadata (i.e. number of words)?
3. Approx. how many data points are required to make an impactful fine-tune?

Q&A

1. In GraphRAG, do we need to rebuild the entire knowledge graph when adding an amendment that changes lease properties?
2. In RAPTOR, would repeating and multi-step retrieval impact performance, especially for thousands of documents?
3. For query expansion, how many semantically similar queries should we generate?