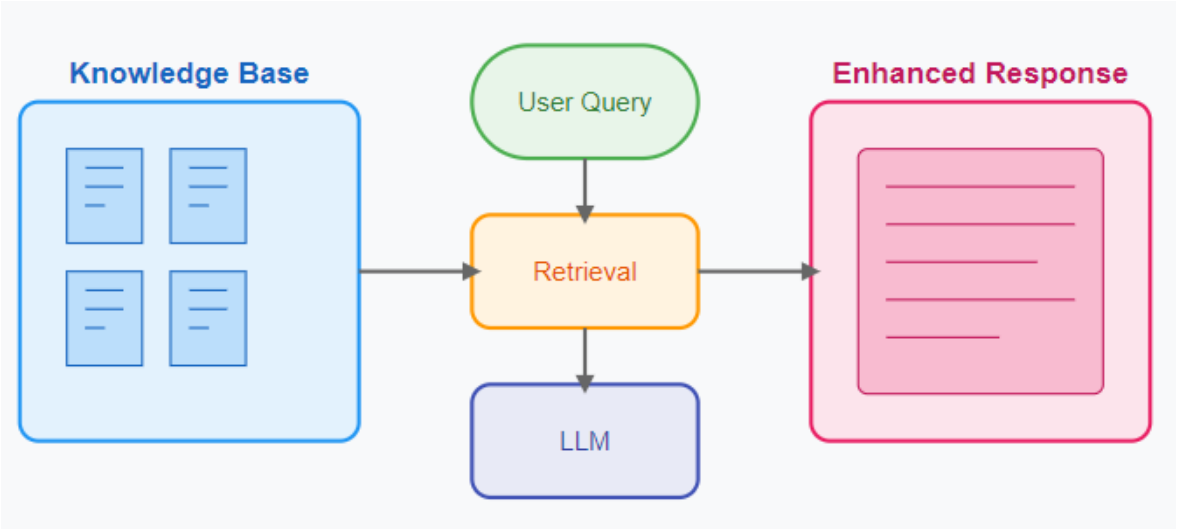
The RAG Approach to Smarter Language Models



About Me!

- Ph.D. entitled “An efficient framework for processing and analyzing unstructured text to discover delivery delay and optimization of route planning in realtime”
- Post-doc at CEA (2019 – 2021) specialized in online machine learning, incremental learning, concept drift, etc.
- Presented by Mohammad ALSHAER, software/data engineer at Meltwater (2022 – present)

الجامعة اللبنانية
UNIVERSITE LIBANAISE

Université Claude Bernard Lyon 1

FROM RESEARCH TO INDUSTRY
cea tech list

Meltwater

Outline

- Introduction to RAG
- RAG Architecture
- Vector Embeddings & Search
- Building a RAG System
- Advanced RAG Techniques
- Evaluation Metrics
- Emerging RAG Research
- Demo

Introduction to RAG

What is RAG?

- Retrieval-Augmented Generation: A hybrid AI architecture that combines retrieval systems with generative models
- First introduced by Lewis et al. in 2020 (Facebook AI Research)[1]
- Enhances Large Language Models (LLMs) with external knowledge retrieval

Why RAG Matters?

- Addresses hallucination issues in LLMs
- Provides up-to-date information beyond training cutoff
- Enables domain-specific knowledge integration
- Improves factual accuracy and citation capabilities

[1] Lewis, Patrick, et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks." *Advances in neural information processing systems* 33 (2020): 9459-9474.

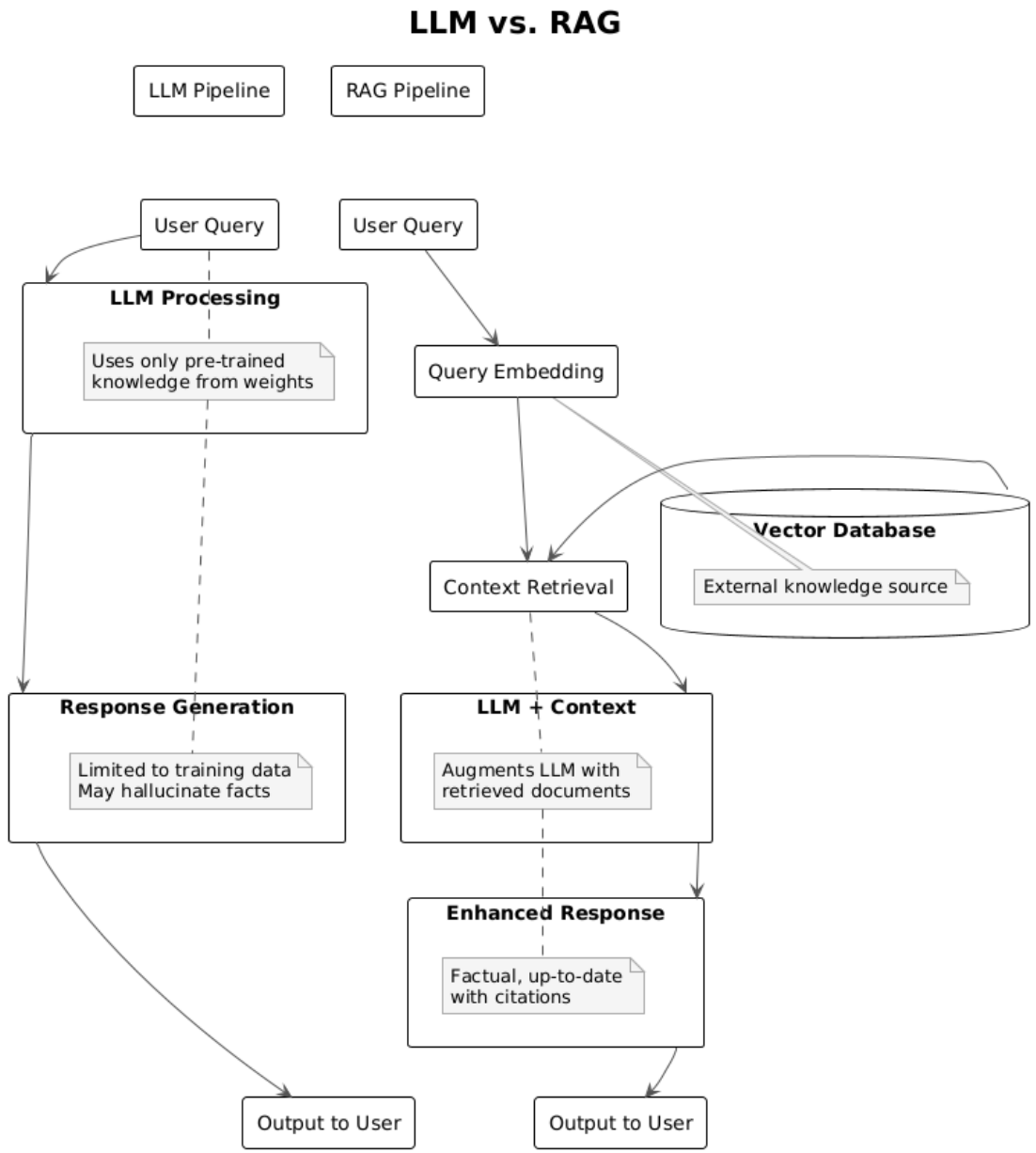
The Problem RAG Solves

Limitations of Traditional LLMs

- Fixed knowledge frozen at training time
- Hallucinations when answering specific queries
- No access to proprietary or specialized information
- Limited context window size (GPT 3.5 had ~4K tokens)

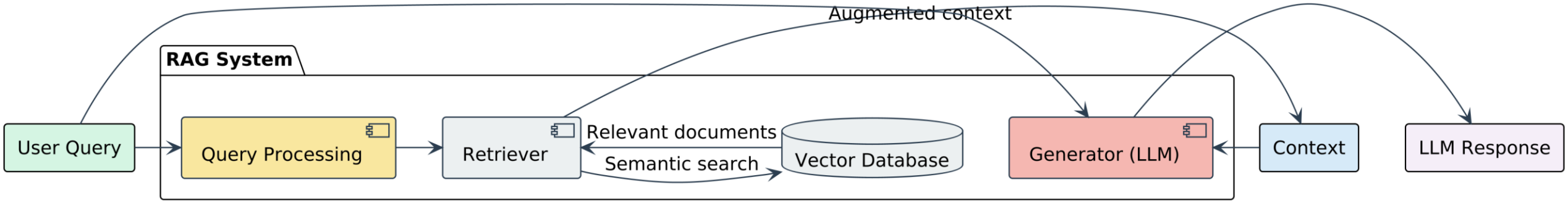
RAG Benefits

- Augments model with real-time external knowledge
- Provides verifiable sources and citations
- Customizable knowledge base for domain adaptation
- Reduces operational costs compared to full retraining



RAG Architecture

- **Chunker**: Breaks documents into manageable pieces
- **Embedder**: Converts text chunks into vector representations
- **Vector Store**: Database optimized for similarity search
- **Retriever**: Fetches relevant documents based on query similarity
- **Generator**: LLM that produces responses based on retrieved context



Vector Embeddings

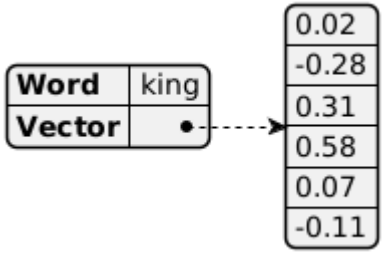
What are Text Embeddings?

- Dense numerical representations of text in high-dimensional space
- Words/phrases with similar meanings are positioned closer together
- Typical dimensions range from 384-1536 depending on the model

Common Embedding Models

- OpenAI's text-embedding-ada-002 (1536d)
- BERT/Sentence-BERT variants (768d)
- BGE embeddings
- E5 embeddings
- Instructor embeddings (customizable for specific tasks)

Vector Embeddings



In reality, embeddings exist in hundreds of dimensions, not just 2

Vector Search

Similarity Metrics

- Cosine Similarity: Measures angle between vectors (scale-invariant)
- Euclidean Distance: Direct distance between points
- Dot Product: Simple multiplication of vector elements

Approximate Nearest Neighbor (ANN) Algorithms

- HNSW (Hierarchical Navigable Small World)
- IVF (Inverted File Index)
- PQ (Product Quantization) for memory efficiency
- FAISS, Annoy, ScaNN, NMSLIB implementations

Vector Search

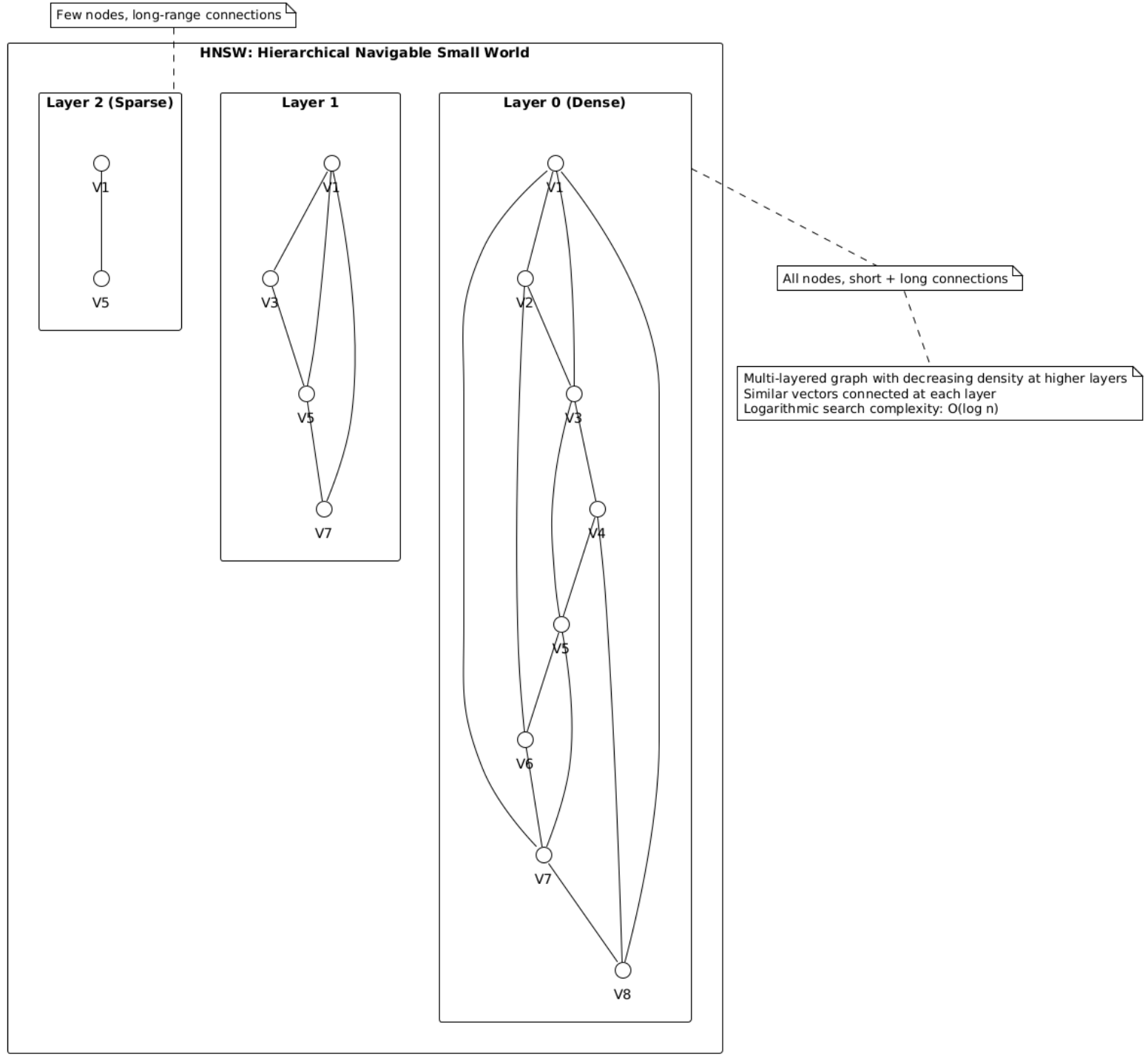
- Nearest Neighbor Search in Action

https://projector.tensorflow.org/

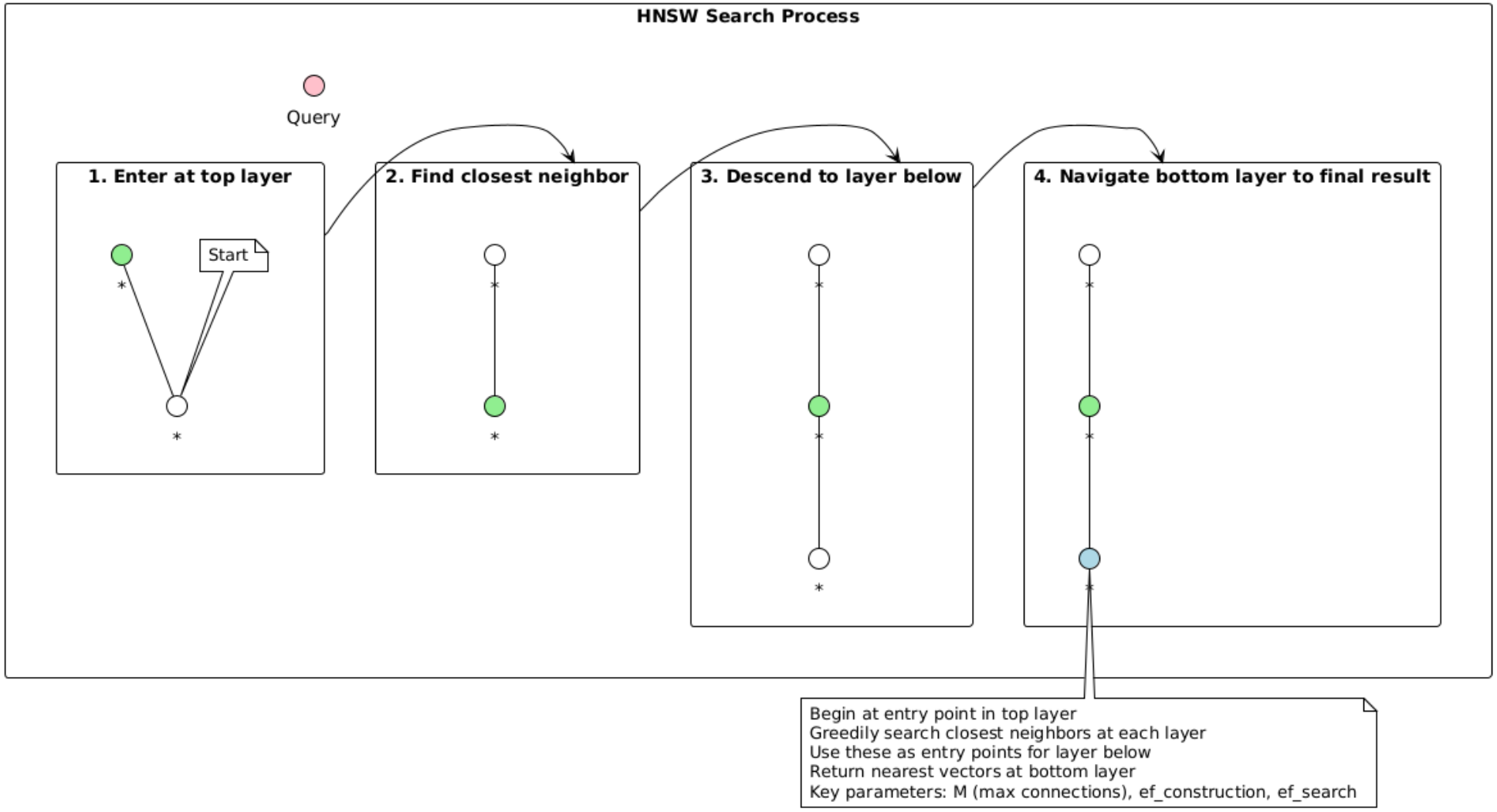
Top 3 nearest neighbors: king, queen, royal

HNSW

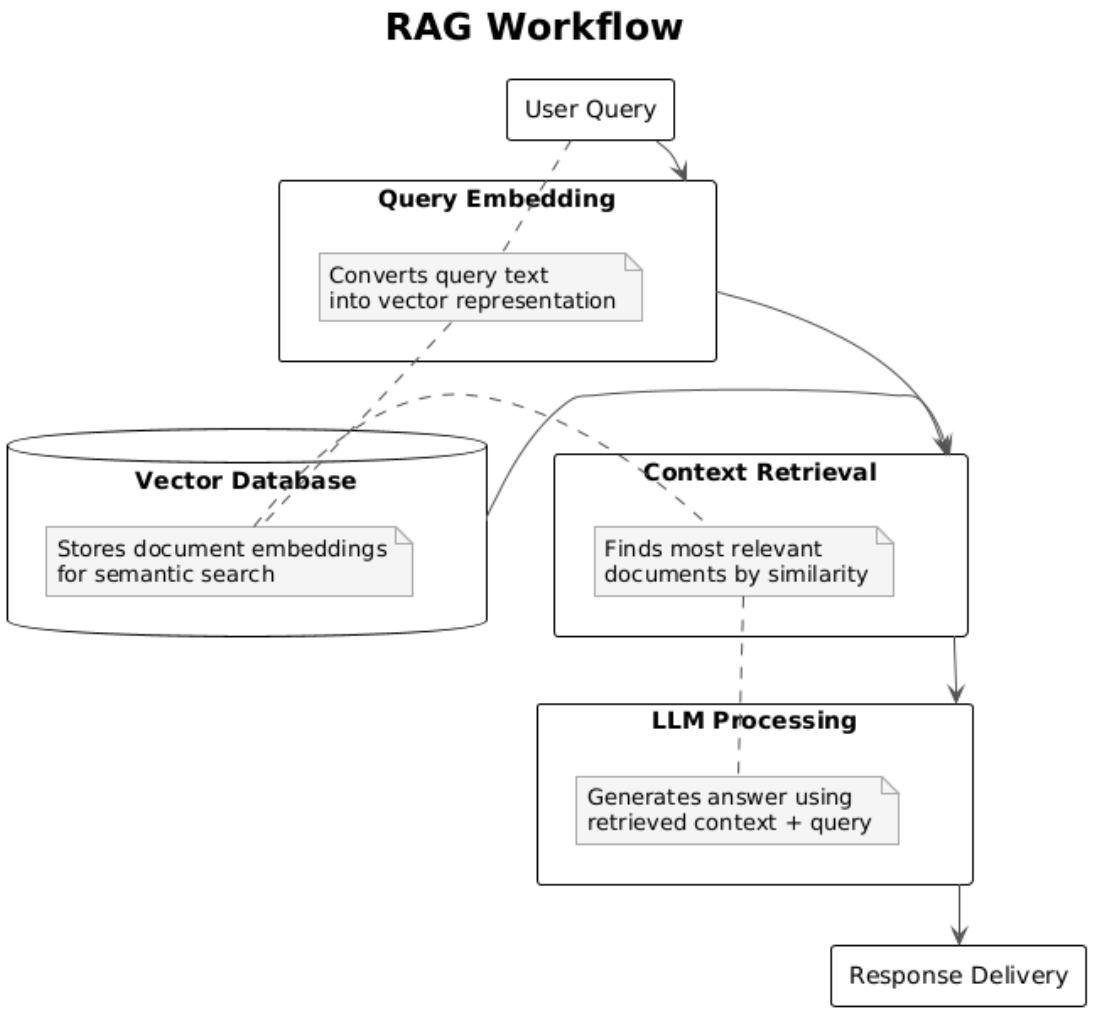
- An efficient graph-based algorithm for ANN search
- Used in vector databases, recommendation systems, and similarity search applications.

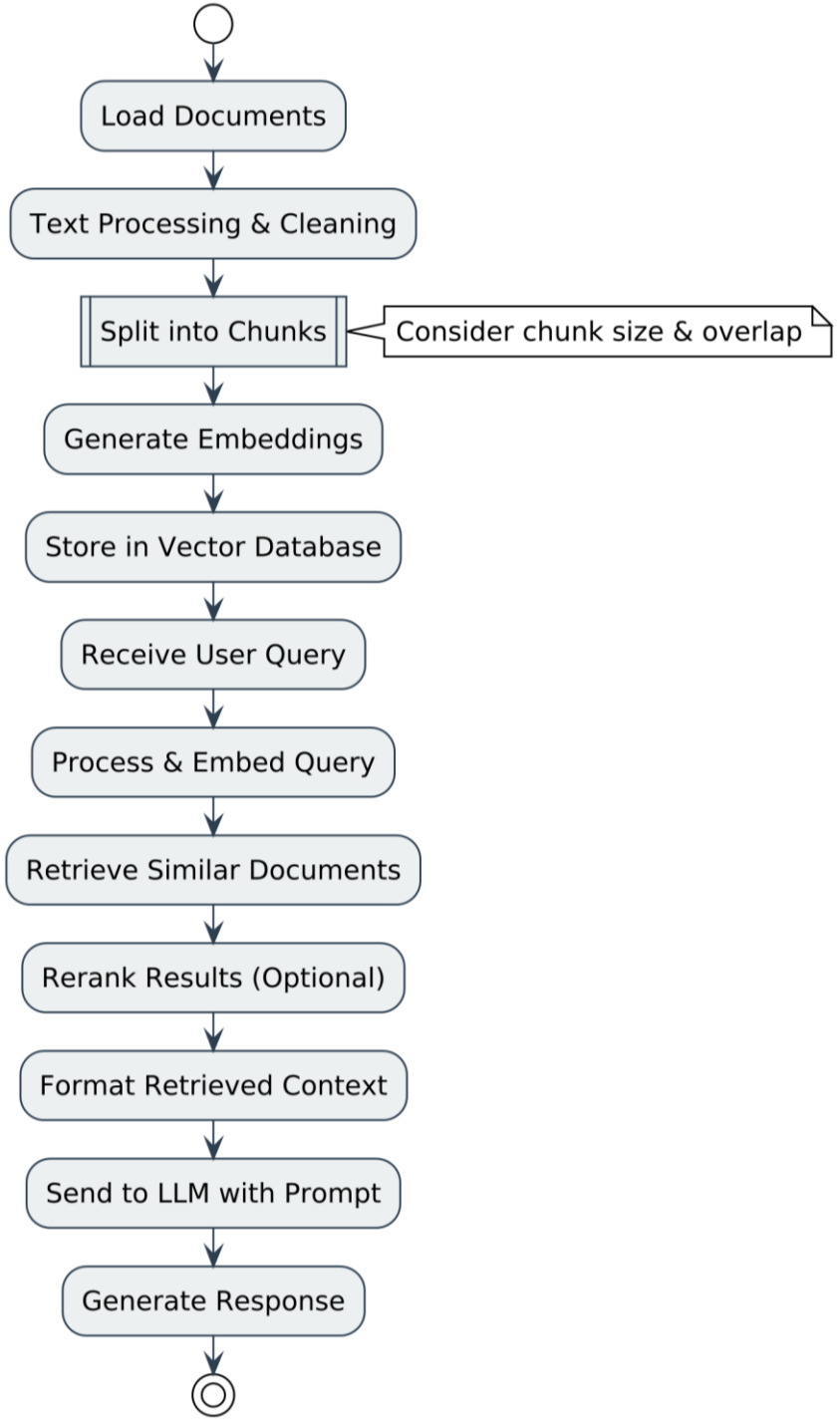


HNSW Search Process



Building a RAG System





RAG Implementation Steps

Document Processing

- Document loading from various sources (PDF, web, databases)
- Text extraction and cleaning
- Chunking strategies (fixed size, semantic, recursive)

Retrieval Methods

- Top-k retrieval
- Hybrid search (lexical + semantic)
- Re-ranking for relevance

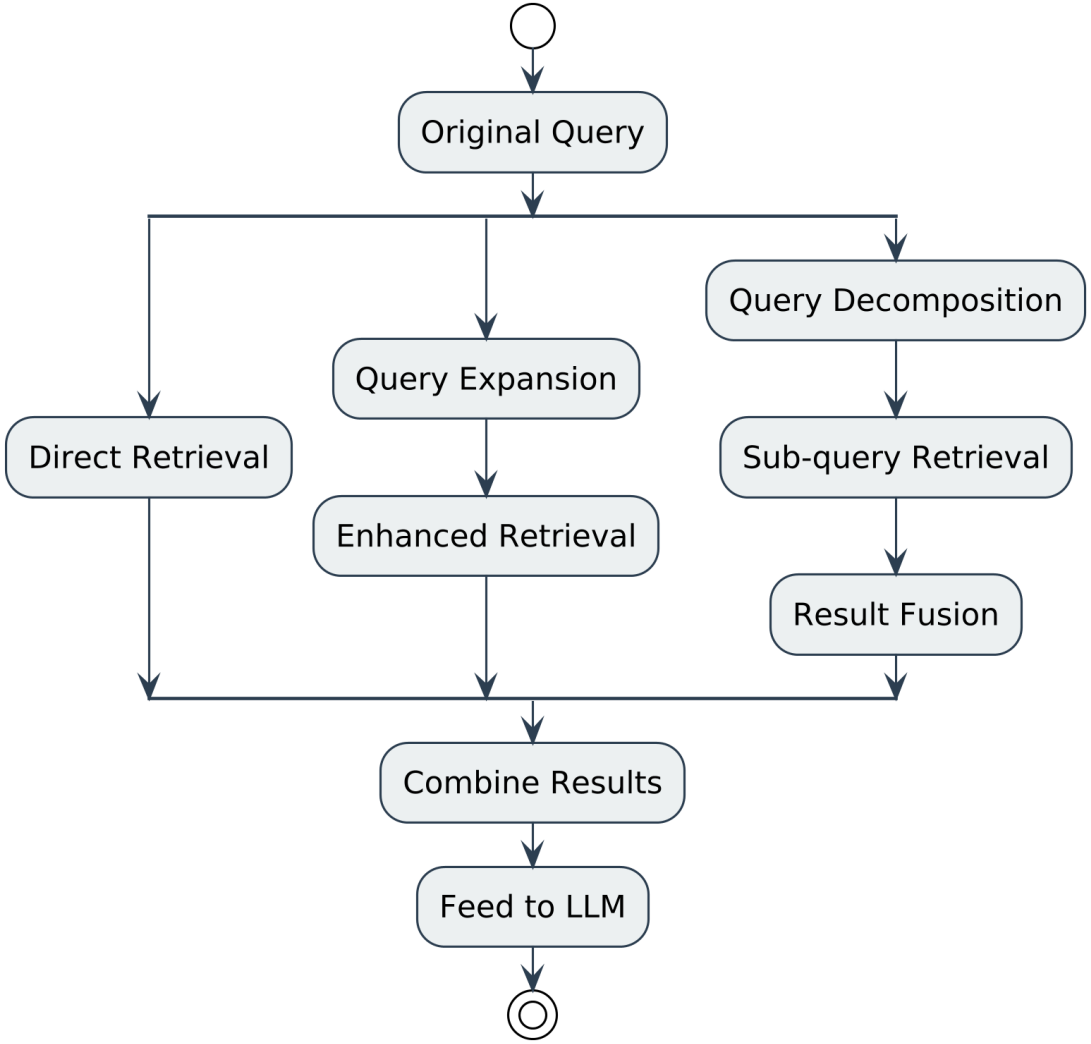
Prompting Strategies

- Few-shot examples
- Instruction formatting
- Source attribution templates

Advanced RAG Techniques (1/2)

Query Transformation

- Query Expansion: Enhancing queries with related terms
- Query Decomposition: Breaking complex queries into sub-queries
- Hypothetical Document Embeddings: Creating ideal document representations



Advanced RAG Techniques (2/2)

Contextual Compression

- Filtering irrelevant information from retrieved documents
- Document summarization before LLM processing

Ensemble Retrieval

- Multiple embedding models
- Multiple retrieval strategies
- Fusion methods for combining retrieval results

Self-RAG and Adaptive Retrieval

- Dynamic retrieval based on uncertainty
- LLM decides when to retrieve information
- Multi-step reasoning with iterative retrieval

Evaluation Metrics

Content Quality

- Factual Correctness: Accuracy of retrieved information
- Hallucination Rate: Measure of fabricated content
- Answer Relevance: Relationship to original query

Retrieval Performance

- Recall: Proportion of relevant documents retrieved
  - Recall@k = (Number of relevant documents retrieved in top k) / (Total number of relevant documents)
- Precision: Proportion of retrieved documents that are relevant
  - Precision@k = (Number of relevant documents in top k) / k

RAG Vector Database Options

Popular Vector DB Solutions

- **Pinecone**: Fully managed vector database service
- **Weaviate**: Open-source vector search engine
- **Milvus**: Distributed vector database
- **Chroma**: Open-source embedding database
- **FAISS**: Facebook AI Similarity Search (library)
- **Qdrant**: Vector search engine
- **pgvector**: Vector extension for PostgreSQL

Selection Factors

- Scale requirements
- Update frequency
- Hosting preferences (cloud vs. on-prem)
- Filtering & metadata capabilities

Common RAG Challenges

Technical Challenges

- Chunking strategy optimization
- Embedding model selection
- Retrieval latency vs. accuracy
- Token context limitations

Implementation Challenges

- Source attribution
- Handling contradictory information
- Multi-hop reasoning
- Domain adaptation

Vector Search Optimization

Indexing Techniques

- Clustering: Group similar vectors
- Quantization: Reduce vector precision for efficiency
- Sharding: Distribute vectors across multiple servers

Performance Tips

- Dimensionality reduction when appropriate
- Pre-filtering based on metadata
- Optimized vector storage formats
- Batch processing for embeddings generation

Emerging RAG Research

- Multimodal RAG: Including images, audio in retrieval
- Recursive Retrieval: Multi-step retrieval processes
- LLM Evaluation of Retrieved Content: Self-critiquing retrieval
- Personalized RAG: User-specific knowledge adaptation

Practical Applications

Industry Use Cases

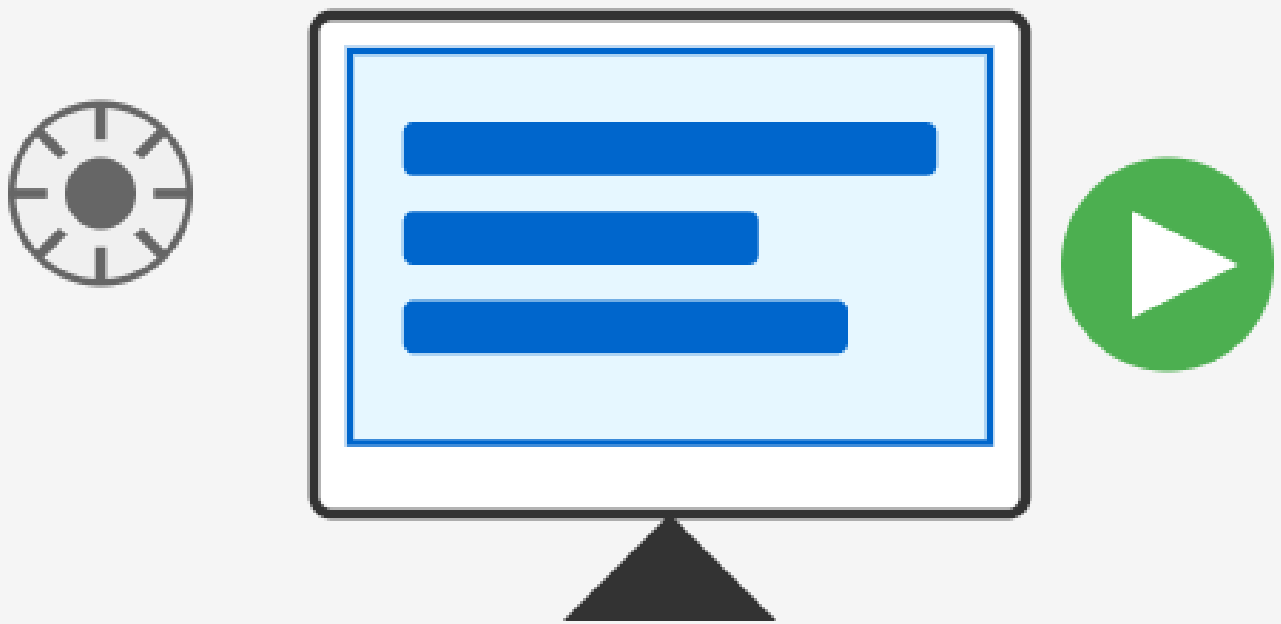
- Customer support knowledge bases
- Legal and medical document analysis
- Educational content personalization
- Enterprise knowledge management

Useful Resources

Tools & Libraries

- LangChain, LlamaIndex (Python frameworks)
- Haystack, Jina AI (RAG pipelines)
- Hugging Face Sentence Transformers (embeddings)

DEMO



Thank you for your attention!