What is Retrieval-Augmented Generation (RAG)?

Retrieval-Augmented Generation (RAG) is a technique that enhances Large Language Models (LLMs) by providing them with relevant external knowledge retrieved from a document store at query time.

Why RAG?
- LLMs have a knowledge cutoff date and can hallucinate
- Fine-tuning is expensive and doesn't scale well for frequently updated data
- RAG provides grounded, up-to-date, and verifiable answers

The RAG Pipeline:
1. Document Ingestion: Load documents (PDF, HTML, text, etc.)
2. Text Extraction: Parse documents into plain text
3. Chunking: Split text into manageable pieces (typically 200-1000 chars)
4. Embedding: Convert chunks into dense vector representations
5. Indexing: Store vectors in a vector database with metadata
6. Query Processing: Convert user query into a vector
7. Retrieval: Find the most similar chunks using vector similarity search
8. Augmentation: Combine retrieved chunks with the user's query
9. Generation: LLM generates an answer grounded in the retrieved context

Key Design Decisions:
- Chunk size: Smaller chunks are more precise but may lack context
- Overlap: Overlapping chunks prevent information loss at boundaries
- Top-K: Number of chunks to retrieve (typically 3-10)
- Similarity metric: Cosine similarity is most common for normalized vectors

Vector Databases and Embedding Models

Vector databases are specialized systems for storing and querying high-dimensional vectors efficiently. They enable fast similarity search over millions of vectors.

Popular Vector Databases:
- pgvector: PostgreSQL extension, great for existing Postgres users
- Pinecone: Fully managed cloud service
- Weaviate: Open-source with hybrid search capabilities
- Qdrant: Open-source with advanced filtering
- ChromaDB: Lightweight, good for prototyping
- Milvus: Open-source, designed for billion-scale vectors

pgvector Index Types:
- IVFFlat: Inverted file index, faster build time, good for smaller datasets
- HNSW: Hierarchical Navigable Small World, better recall, recommended for production use. Supports cosine, L2, and inner product distances.

Embedding Models:
The choice of embedding model significantly affects retrieval quality.

Key considerations:
- Dimension size: Higher dimensions capture more nuance (512-4096 typical)
- Training data: Models trained on similar domains perform better
- Asymmetric models: Use different encoders for queries vs passages (e.g., NVIDIA NV-EmbedQA uses input_type "query" vs "passage")
- Multilingual support: Important for non-English documents

Popular models: OpenAI text-embedding-3, Cohere Embed v3, NVIDIA NV-EmbedQA, BGE, E5, and sentence-transformers models on HuggingFace.

Advanced RAG Techniques

Beyond basic RAG, several techniques can improve retrieval quality and answer accuracy.

Hybrid Search:
Combine dense vector search with traditional keyword search (BM25). This captures both semantic similarity and exact keyword matches.

Re-ranking:
After initial retrieval, use a cross-encoder model to re-rank results. Cross-encoders are more accurate than bi-encoders but slower, so they are applied only to the top-K candidates.

Query Transformation:
- Query expansion: Add related terms to improve recall
- HyDE (Hypothetical Document Embeddings): Generate a hypothetical answer first, then use its embedding for retrieval
- Multi-query: Generate multiple query variations and merge results

Chunking Strategies:
- Fixed-size: Simple, consistent chunk sizes
- Sentence-based: Split on sentence boundaries
- Semantic: Use embedding similarity to find natural breakpoints
- Recursive: Try multiple separators (paragraphs, sentences, words)
- Parent-child: Store small chunks for retrieval but return larger parent chunks for context

Metadata Filtering:
Store metadata (source, date, author, topic) with chunks and use it to filter results before or after vector search.

Evaluation:
Use frameworks like RAGAS to measure:
- Faithfulness: Is the answer grounded in the retrieved context?
- Relevancy: Are the retrieved chunks relevant to the query?
- Context precision: How precise is the retrieval?
- Answer correctness: Is the final answer correct?