Pre-Read Document

Retrieval-Augmented Generation (RAG), Embeddings, and Vector Search Systems

This document provides foundational preparation for sessions covering RAG, embeddings, vector databases, and retrieval pipeline design. It introduces key concepts required to understand modern retrieval-augmented AI systems and prepares participants for deeper architectural discussions.

---

1. What is Retrieval-Augmented Generation (RAG)?

RAG combines:

- Information retrieval
- Large language models

Instead of relying only on model training data, RAG retrieves relevant external documents at runtime and provides them as context to the model before generation.

Core Idea:

Ground the model's response using external knowledge.

1.1 Why RAG?

RAG is useful when:

- Knowledge changes frequently
- Domain-specific data is large
- Transparency and citations are required
- Hallucination reduction is critical

---

2. Embeddings

2.1 What Are Embeddings?

Embeddings are dense numerical vector representations of data (text, images, etc.) such that semantically similar inputs are geometrically close in vector space.

Example:
Two similar sentences will have vectors close to each other.

2.2 Token vs Sentence vs Document Embeddings

Token Embeddings

- Represent individual words or subwords
- Used internally inside Transformers
- Contextual in modern LLMs

Sentence Embeddings

- Represent entire sentences
- Used in semantic similarity and retrieval

Document Embeddings

- Represent larger text chunks
- Often computed by pooling sentence or token embeddings

Choice depends on retrieval granularity and use case.

---

3. Vector Space Intuition

Embeddings exist in high-dimensional vector space.

Key ideas:

- Each dimension encodes learned semantic properties
- Distance reflects similarity
- Clusters form around related meanings

Visualization (conceptual):

Points closer together → higher semantic similarity.

---

4. Semantic Similarity

Semantic similarity measures meaning similarity rather than exact word matching.

Example:

- “Car insurance policy”
- “Vehicle coverage document”

Keyword match is low. Semantic similarity is high.

This enables more intelligent retrieval than traditional keyword search.

---

5. Similarity and Distance Metrics

To compare vectors, we use mathematical distance measures.

5.1 Cosine Similarity

Measures angle between vectors:

$\cos(\theta) = (A \cdot B) / (||A|| \ ||B||)$

Values range from:

- 1 → identical direction
- 0 → orthogonal
- -1 → opposite direction

Most commonly used in text retrieval.

5.2 Other Distance Metrics

- Euclidean distance
- Dot product
- Manhattan distance

Choice depends on embedding normalization and system design.

---

6. Embedding Quality

Embedding quality determines retrieval effectiveness.

Factors influencing quality:

- Model architecture
- Training dataset
- Domain alignment
- Chunk size
- Text preprocessing

Evaluation metrics:

- Retrieval recall

- Precision
- Semantic clustering quality

---

7. Multimodal Embeddings

Modern systems can embed:

- Text
- Images
- Audio

These embeddings exist in shared vector spaces.

Applications:

- Text-to-image retrieval
- Image caption search
- Cross-modal semantic search

---

8. Vector Databases

8.1 Why a Vector Database is Needed

Traditional databases are optimized for:

- Exact matches
- Structured queries

They are not optimized for:

- High-dimensional nearest neighbor search

Vector databases are designed for:

- Storing embeddings
- Fast similarity search
- Scalable nearest neighbor retrieval

8.2 Vector Storage

Each stored record includes:

- Vector embedding
- Metadata (source, section, timestamp, tags)

This allows filtering before or after similarity search.

---

9. Approximate Nearest Neighbor (ANN)

Exact nearest neighbor search becomes computationally expensive at scale.

ANN algorithms trade slight accuracy loss for massive performance gains.

Benefits:

- Faster search
- Scalable to millions or billions of vectors
- Suitable for real-time systems

9.1 HNSW (Hierarchical Navigable Small World)

Graph-based ANN algorithm.

Characteristics:

- Builds multi-layer graph
- Efficient search traversal
- High recall
- Widely used in production systems

9.2 IVF (Inverted File Index)

Cluster-based approach:

- Partition vector space into clusters

- Search only relevant clusters

Benefits:

- Reduced search space
- Scalable indexing

---

10. Metadata Filtering

Metadata enables structured filtering before or during retrieval.

Example:

- Filter by document type
- Filter by publication date
- Filter by product category

Combining metadata filtering with vector similarity improves relevance.

---

11. Hybrid Search

Hybrid search combines:

- Sparse retrieval (keyword/BM25)

- Dense retrieval (embeddings)

Why it matters:

- Captures exact matches
- Preserves semantic meaning
- Improves recall

---

12. Ranking and Re-Ranking

Initial retrieval may not be optimal.

Re-ranking methods:

- Cross-encoder models
- LLM-based ranking
- Weighted scoring (sparse + dense)

Purpose:

Improve final document ordering before generation.

---

13. RAG Pipeline Overview

High-level flow:

1. User query
2. Query embedding
3. Vector search
4. Retrieve top-k documents
5. Context assembly
6. LLM generation
7. Response

---

14. RAG vs Fine-Tuning

RAG

- Dynamic knowledge
- External document retrieval
- Easy updates
- Transparent sources

Fine-Tuning

- Knowledge embedded in weights

- Harder to update
- Good for behavior/style control

Hybrid systems often combine both approaches.

---

15. Developing RAG-Based Applications

15.1 Core Development Steps

1. Document ingestion
2. Cleaning and preprocessing
3. Chunking
4. Embedding generation
5. Vector storage
6. Retrieval pipeline design
7. Context assembly
8. LLM integration
9. Evaluation

15.2 Chunking Strategies

- Fixed-size chunks
- Overlapping chunks
- Semantic splitting
- Section-based splitting

Chunk size affects:

- Retrieval precision
- Context completeness
- Token usage

---

16. RAG Data Ingestion

Data ingestion determines retrieval performance.

Key stages:

1. Data collection
2. Normalization
3. Chunking

4. Embedding
5. Indexing
6. Metadata tagging

Poor ingestion leads to poor retrieval.

---

17. Evaluation of RAG Systems

Evaluate both:

Retrieval:

- Recall@k
- Precision@k

Generation:

- Faithfulness
- Groundedness
- Hallucination rate

Continuous evaluation improves reliability.

---

18. Advanced Directions for Exploration

- Multi-hop retrieval
- Query rewriting
- Context compression
- Retrieval confidence scoring
- Semantic caching
- Graph-augmented retrieval
- Long-context LLM integration

Preparation Before the Session

Participants should be comfortable with:

- Basic linear algebra concepts
- Understanding of embeddings
- Similarity metrics
- Fundamentals of LLM operation
- Differences between retrieval and training

This preparation ensures deeper architectural discussions can focus on trade-offs, optimization strategies, and real-world implementation considerations rather than basic definitions.