Retrieval-Augmented Generation (RAG) – Interview Q&A

Q1. What is RAG?
A: RAG = Retrieval-Augmented Generation.
It combines **retrieval** (finding relevant info from external knowledge like vector DBs) with **generation** (LLM producing an answer).

- Retrieval ensures the model has access to updated, factual info.
- Generation ensures natural language responses.

Q2. Why do we need RAG instead of just fine-tuning an LLM?
A:

- Fine-tuning requires retraining when knowledge updates → costly.
- RAG can plug into live databases → always up-to-date.
- RAG handles domain-specific/private data without retraining the LLM.

Q3. What are the main steps in a RAG pipeline?
A:

1. **Ingestion** – Load documents (PDF, Word, HTML).
2. **Chunking** – Split into smaller parts (e.g., 300 words).
3. **Embedding** – Convert chunks into vector representations (using models like `all-MiniLM`).
4. **Indexing** – Store embeddings in a vector DB (FAISS, Pinecone, Chroma).
5. **Query Processing** – Convert user query into vector.
6. **Retrieval** – Search top-k relevant chunks.
7. **Generation** – LLM generates an answer using retrieved chunks + query.

Q4. What is an embedding model?
A: A model that converts text into **numerical vectors**.
These vectors capture **semantic meaning**, not just keywords.
Example: *"Physics is my favorite subject"* → `[0.6, 0.3, 0.1, ...]`

Q5. What is FAISS?
A: FAISS (Facebook AI Similarity Search) is a library for:

- Storing embeddings
- Indexing vectors
- Performing **fast similarity search** using cosine similarity or Euclidean distance.

Q6. What retrieval strategies are used in RAG?
A:

- **Top-k retrieval** → Return top N similar chunks.
- **Metadata filters** → Filter by tags/date/source.

Q7. What are the advantages of RAG?
A:

- Reduces hallucinations by grounding answers in data.
- Handles private/custom knowledge bases.
- No retraining required for new data

Q8. How does RAG reduce hallucinations?

A: By grounding LLM outputs in **retrieved factual data**.

Instead of generating answers from internal parameters only, it **cites external documents** increasing factual accuracy.

Q9. What vector databases are commonly used in RAG?
A:

- **FAISS** (open-source, fast, local).
- **Pinecone** (cloud, scalable).
- **ChromaDB** (lightweight, open-source).
-

Q10. What is the role of similarity metrics in RAG?
A: To compare **query embeddings** vs **document embeddings**.

- **Cosine similarity** → semantic closeness (most common).
- **Euclidean distance** → geometric closeness in vector space.

Q11. Can RAG handle multimodal data (text + images + audio)?

A: Yes, if embeddings are multimodal.

Example: CLIP for text–image embeddings → store both in vector DB → retrieval works across modalities.

Q12. Explain a real-world use case of RAG.
A:

- **Enterprise chatbot** → Uses RAG to answer employee queries from internal PDFs.
- **Customer support bot** → Retrieves policies/FAQs.
- **Healthcare assistant** → Answers using medical research papers.

Q13. What are the main challenges with RAG?
A:

- **Chunking strategy** → Too small = loss of context, too large = poor retrieval.
- **Embedding quality** → Weak embeddings → irrelevant retrieval.
- **Latency** → Retrieval + LLM call adds overhead.
- **Hallucinations** → If retrieved chunks are weak, model may still hallucinate.
- **Cost** → Embedding + vector DB + LLM inference = expensive at scale.

Q14. How do you choose the chunk size in RAG?
A:

- Typical size: **200–500 tokens** (depends on model context window).
- **Too small** → retrieval may miss context.
- **Too large** → retrieval less precise, embedding inefficiency.
- Often tuned based on **domain** + **retrieval performance**

Q15. How do you reduce latency in a RAG pipeline?
A:

- Use **smaller embedding models** (MiniLM vs large BERT).
- Cache frequent queries.
- Retrieve fewer chunks (top-3 instead of top-10).

Q16. Compare RAG with Retrieval-only systems.
A:

- **Retrieval-only** → returns passages (user must read).
- **RAG** → synthesizes passages into final natural-language answer.
- RAG = more user-friendly

Q17. How do you improve retrieval accuracy in RAG?
A:

- Better embedding model (e.g., OpenAI `text-embedding-3-large`).
- Hybrid search (BM25 + semantic).
- Re-ranking using cross-encoders (BERT-based rerankers).
- Domain-specific fine-tuned embeddings.