🎓 RAG Systems Comprehensive Quiz

Retrieval-Augmented Generation Assessment

20 Multiple Choice Questions | Mixed Difficulty

Test your understanding of RAG concepts, implementation, and best practices

---

Question 1 **Easy**

What is the primary purpose of Retrieval-Augmented Generation (RAG)?

A To make LLMs generate text faster

B To ground LLM responses in external knowledge and reduce hallucination

C To train LLMs with less data

D To compress LLM model size

✓ Correct Answer: B

RAG's primary purpose is to **ground LLM responses in external knowledge sources** and reduce hallucination. By retrieving relevant documents before generation, RAG ensures answers are based on actual information rather than the model's parametric knowledge alone. This addresses key LLM limitations: knowledge cutoffs, inability to access private data, and tendency to confidently generate false information.

Question 2 Medium

In the RAG pipeline, what is the purpose of the "chunking" step?

A To compress documents to save storage space

B To split large documents into smaller, semantically meaningful pieces for better retrieval

C To remove unnecessary words from documents

D To translate documents into multiple languages

✓ Correct Answer: B

Chunking splits large documents into **smaller, semantically meaningful pieces**. This is crucial because: (1) embeddings work better on focused text, (2) retrieval becomes more precise when matching specific segments, (3) LLMs have context window limits, and (4) smaller chunks improve the signal-to-noise ratio in retrieved context. The goal is to create retrievable units that contain coherent, self-contained information.

Question 3 Hard

Which chunking strategy would be MOST appropriate for a legal document database where precise citations to specific paragraphs are critical?

A Fixed-size chunking with 500 characters

B Semantic chunking based on topic shifts

C Paragraph-based chunking that preserves document structure

D Sentence-based chunking with 10-sentence groups

✓ Correct Answer: C

Paragraph-based chunking is ideal for legal documents because: (1) legal texts are structured with meaningful paragraph boundaries, (2) citations in law reference specific paragraphs or sections, (3) preserving this structure maintains legal meaning and allows accurate source attribution, (4) legal paragraphs are typically self-contained units of meaning. Fixed-size would break mid-paragraph, semantic chunking might merge paragraphs incorrectly, and sentence-based would lose the natural legal structure.

Question 4 **Easy**

What are embeddings in the context of RAG systems?

A Compressed versions of text files

B Dense numerical vectors that capture semantic meaning of text

C Hyperlinks between documents

D Metadata tags for categorizing documents

✓ Correct Answer: B

Embeddings are **dense numerical vectors** (typically 384 to 3072 dimensions) that represent text in a high-dimensional space where semantic similarity corresponds to proximity. Unlike sparse representations (like TF-IDF), embeddings capture meaning—"dog" and "puppy" have similar vectors even though they don't share letters. This enables semantic search where queries match based on meaning, not just keywords.

Question 5 Medium

Why is cosine similarity preferred over Euclidean distance for comparing text embeddings?

A It's faster to compute

B It's invariant to vector magnitude and only measures direction (meaning)

C It works better with high-dimensional spaces

D It produces values between 0 and 1

✓ Correct Answer: B

Cosine similarity is preferred because it's **invariant to vector magnitude**—it only measures the angle (direction) between vectors, not their length. This is crucial for text because: a longer document shouldn't automatically be "more similar," we care about semantic orientation not size, and normalized embeddings focus purely on meaning. Cosine ranges from -1 to 1 (though text embeddings typically yield 0 to 1), whereas Euclidean distance is unbounded and magnitude-dependent.

Question 6 **Hard**

A RAG system retrieves documents with similarity scores of 0.45, 0.52, and 0.48. What should you do?

- **A** Use all three documents as they're the top results
- **B** Set a higher similarity threshold (e.g., 0.6) and inform the user no relevant documents were found
- **C** Increase the chunk size and retry
- **D** Use a different embedding model

✓ Correct Answer: B

Scores around **0.45-0.52 indicate weak relevance**—these documents are marginally related at best. Using them would likely result in unhelpful or misleading answers. The best practice is to **set a minimum similarity threshold** (typically 0.6-0.7) and gracefully handle the "no relevant documents" case by informing the user rather than generating answers from poor matches. This prevents hallucination and maintains system credibility. Options C and D might help long-term but don't address the immediate low-quality retrieval.

Question 7 Easy

What is the role of a vector database in RAG systems?

- **A** To store the original text documents
- **B** To efficiently store embeddings and perform fast similarity search
- **C** To generate embeddings from text
- **D** To train the language model

✓ Correct Answer: B

Vector databases are specialized for **storing high-dimensional vectors and performing fast similarity search**. Traditional databases aren't optimized for operations like "find the 5 most similar vectors to this query vector" across millions of vectors. Vector DBs use algorithms like HNSW, IVF, or LSH to enable sub-linear search (O(log n) instead of O(n)). They store embeddings with metadata and enable retrieval in milliseconds even with large datasets.

Question 8 Medium

Which scenario would benefit MOST from RAG rather than fine-tuning?

A Teaching an LLM to write in a specific literary style

B Answering questions about a frequently updated product catalog

C Generating creative fiction in a fantasy world

D Translating between two specific languages

✓ Correct Answer: B

RAG excels when dealing with **frequently changing information** like product catalogs. Fine-tuning bakes knowledge into model weights, requiring expensive retraining for updates. RAG simply updates the document store—instant refresh without retraining. Option A (style) and D (translation) are behavior changes best suited for fine-tuning. Option C (creative fiction) doesn't need external knowledge retrieval. RAG is ideal for: dynamic knowledge, large document collections, need for citations, and private/proprietary data.

Question 9 Hard

You're building a RAG system with 100,000 document chunks. Which indexing algorithm should you use for optimal performance?

A Flat (brute force) indexing for perfect accuracy

B HNSW (Hierarchical Navigable Small World) for sub-linear search

C No indexing, just store in a regular SQL database

D Hash-based indexing for O(1) lookup

✓ Correct Answer: B

HNSW (Hierarchical Navigable Small World) is ideal for 100K vectors because: (1) it provides O(log n) search complexity vs O(n) for brute force, (2) achieves ~99% recall with 100-1000x speedup, (3) scales well to millions of vectors, (4) commonly used in production vector DBs like Pinecone and Weaviate. Flat indexing would work but be unnecessarily slow (~100K comparisons per query). SQL databases can't efficiently search high-dimensional vectors. Hash-based indexing doesn't support similarity search (only exact matches).

Question 10 **Medium**

What is the main disadvantage of using very small chunk sizes (e.g., 100 characters)?

A Slower embedding generation

B Chunks lack sufficient context, leading to poor answer quality

C Higher storage costs

D Incompatible with most embedding models

✓ Correct Answer: B

Very small chunks **lack sufficient context** for meaningful answers. A 100-character chunk might be a sentence fragment that can't stand alone. While retrieval precision might be high (you find the exact relevant sentence), the retrieved context doesn't contain enough information for the LLM to generate a complete answer. This is the classic **precision-recall tradeoff**: smaller chunks = better precision but worse recall of full context. Typical sweet spot is 500-1000 characters.

Question 11 Easy

What does "top-K" mean in the context of RAG retrieval?

A The K most important keywords in a document

B The K highest-scoring (most similar) chunks retrieved

C The K longest documents in the database

D The K most recent documents added

✓ Correct Answer: B

Top-K retrieval means retrieving the K documents/chunks with the highest similarity scores to the query. For example, top-3 retrieval returns the 3 most similar chunks. This is a fundamental parameter in RAG: too low (K=1) and you might miss relevant info; too high (K=20) and you add noise and exceed LLM context limits. Typical values are K=3-5 for most applications. The "K" is adjustable based on your precision-recall requirements.

Question 12 **Hard**

Your RAG system is hallucinating despite retrieving relevant documents. What is the MOST likely cause?

A The embedding model is too small

B The prompt doesn't strongly instruct the LLM to use only the provided context

C The vector database is configured incorrectly

D The chunk size is too large

✓ Correct Answer: B

If retrieval is working (relevant docs are found) but the LLM still hallucinates, the issue is in the **generation phase—specifically prompt engineering**. The LLM needs explicit, strong instructions like "Answer ONLY based on the provided context" or "If the answer is not in the context, say 'I don't know'". Without these constraints, even with good retrieval, the LLM may rely on its parametric knowledge and generate information not in the retrieved docs. Temperature should also be low (0.1-0.3) for factual responses.

Question 13 Medium

What is the purpose of chunk overlap in document processing?

- **A** To duplicate data for redundancy
- **B** To prevent information loss at chunk boundaries
- **C** To increase the total number of chunks
- **D** To compress the documents more efficiently

✓ Correct Answer: B

Chunk overlap (typically 10-20% of chunk size) **prevents information loss at boundaries**. Consider a sentence that gets split: first chunk ends mid-sentence, next chunk starts mid-sentence—both chunks lose meaning. Overlap ensures that information near boundaries appears in multiple chunks, so at least one chunk will have complete context. Example: with 500-char chunks and 100-char overlap, the last 100 chars of chunk N appear as the first 100 chars of chunk N+1. This maintains semantic continuity.

Question 14 Easy

Which of the following is NOT a typical component of a RAG system?

- **A** Embedding model

B Vector database

C Gradient descent optimizer

D Large language model

✓ Correct Answer: C

Gradient descent optimizer is used during model training, not in RAG inference. RAG systems use pre-trained models (embedding model and LLM) without further training. The typical components are: (1) **Embedding model** to convert text to vectors, (2) **Vector database** to store and search embeddings, (3) **LLM** to generate answers from retrieved context, plus document processing and prompt engineering components. RAG is an inference-time technique, not a training method.

Question 15 **Hard**

A company wants to build a RAG system that must work offline on employee laptops. Which architecture should they choose?

A OpenAI embeddings + Pinecone + GPT-4

B Sentence-transformers + FAISS + local LLM (e.g., Llama)

C Google PaLM embeddings + Weaviate Cloud + Claude

D Cohere embeddings + ChromaDB + DeepSeek

✓ Correct Answer: B

For **offline operation**, you need fully local components with no API calls: (1) **Sentence-transformers** - open-source embedding models that run locally, (2) **FAISS** - local vector search library (no cloud service), (3) **Local LLM** like Llama 2, Mistral, or Phi that can run on laptops. All other options include cloud APIs (OpenAI, Pinecone, Weaviate Cloud, Cohere, DeepSeek) that require internet. ChromaDB in option D can work locally, but Cohere and DeepSeek require API access.

Question 16 **Medium**

What is "semantic chunking" and why might it be superior to fixed-size chunking?

- **A** Chunking based on file size; it's faster
- **B** Chunking that uses embeddings to detect topic shifts and creates natural boundaries; better coherence
- **C** Random chunking; more diverse results
- **D** Chunking by word count; more precise

✓ Correct Answer: B

Semantic chunking uses embeddings to identify where topics change in a document, creating chunks at natural semantic boundaries. Process: (1) split into sentences, (2) embed each sentence, (3) calculate similarity between consecutive sentences, (4) split where similarity drops (topic shift). This is superior to fixed-size because: chunks align with actual topics, no awkward mid-sentence/mid-paragraph splits, each chunk is semantically coherent. Trade-off: computationally expensive (must embed all sentences) and produces variable chunk sizes.

Question 17 Easy

In RAG, what happens during the "retrieval" phase?

A Documents are loaded from disk

B The most similar chunks to the query are found and returned

C The LLM is trained on new data

D Embeddings are generated for all documents

✓ Correct Answer: B

The **retrieval phase** is when the system searches for relevant information to answer a query. Steps: (1) embed the user's query, (2) search vector database for most similar document chunks (using cosine similarity), (3) return top-K chunks with highest similarity scores. This happens at query time (every user question), unlike indexing (done once) or training (not part of RAG). The retrieved chunks then become context for the LLM's answer generation.

Question 18 Hard

Your RAG system must handle documents in both English and Spanish. What should you do?

A Translate all documents to English first

B Use a multilingual embedding model (e.g., multilingual-MiniLM) that maps both languages to the same vector space

C Build separate RAG systems for each language

D Use character-level embeddings

✓ Correct Answer: B

Multilingual embedding models are trained to map text from different languages into a shared semantic space—semantically similar text has similar vectors regardless of language. This means: a Spanish query can retrieve English documents (and vice versa), cross-lingual semantic search works naturally, single vector database for all languages. Models like multilingual-MiniLM, LaBSE, or Cohere's multilingual embeddings support this. Option A (translation) adds latency and errors. Option C (separate systems) fragments knowledge and can't do cross-lingual retrieval.

Question 19 Medium

What is the primary advantage of using metadata filtering in RAG retrieval?

A It makes embeddings smaller

B It narrows the search space to relevant subsets, improving precision and speed

C It eliminates the need for embeddings

D It automatically translates documents

✓ Correct Answer: B

Metadata filtering constrains retrieval to specific document subsets based on attributes like date, category, author, or department. Benefits: (1) **improved precision** - only search relevant documents (e.g., "only 2024 documents"), (2) **faster search** - smaller search space, (3) **business logic** - enforce access control or recency requirements, (4) **context control** - ensure retrieved docs meet query constraints. Example: "What's our Q3 policy?" filters to category="policy" AND quarter="Q3" before semantic search.

Question 20 Hard

Which of these scenarios would benefit LEAST from implementing a RAG system?

A Answering customer questions about a 500-page product manual

B Generating creative fiction stories in a fantasy setting

C Legal research across thousands of case documents

D Technical support chatbot for software documentation

✓ Correct Answer: B

Creative fiction generation benefits least from RAG because: (1) creativity requires imagination, not retrieval of facts, (2) constraining generation to retrieved context limits creative freedom, (3) fiction doesn't need grounding in external documents, (4) the goal is novel content, not accurate information. RAG is designed for **knowledge-intensive tasks** where factual accuracy matters. Options A, C, and D all involve retrieving and accurately representing information from documents—perfect RAG use cases. For fiction, you want an unconstrained LLM or fine-tuning for style.

🎯 Quiz Complete!

Review your answers and explanations above to assess your understanding of RAG systems.

Scoring Guide:
18-20 correct: Expert level 🌟
15-17 correct: Advanced understanding 📚
12-14 correct: Good foundation 👍
9-11 correct: Review key concepts 📖
Below 9: Revisit tutorial materials 🔄